Tiago Miguel Mendes Ferreira Vitorino

Licenciado em Ciências da Engenharia Electrotécnica e de Computadores

Planning and Verification of Multipath Routing Protocols

Dissertação para obtenção do Grau de Mestre em Engenharia Electrotécnica e de Computadores, pela Universidade Nova de Lisboa, Faculdade de Ciências e Tecnologia.

Orientador: Pedro Amaral, Professor Doutor, FCT-UNL

Júri:

Presidente: Doutor André Teixeira Bento Damas Mora – FCT/UNL
Arguente: Doutor Paulo da Costa Luís da Fonseca Pinto – FCT/UNL
Vogal: Doutor Pedro Miguel Figueiredo Amaral – FCT/UNL

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Setembro, 2014

Planning and Verification of Multipath Routing Protocols

Agradecimentos

Este espaço é dedicado a todos aqueles que me acompanharam durante a elaboração desta dissertação. Para todos eles, o meu mais sincero obrigado!

Primeiro, gostaria de agradecer a toda a minha família, e em especial aos meus pais e irmãos pela compreensão, apoio, e muita paciência durante todo este tempo e por me terem dado a oportunidade de atingir este nível académico. À minha namorada, por toda a amizade, amor, dedicação e apoio incondicional ao longo da minha formação. A todos os meus amigos em especial ao David Amoêdo, Pedro Freitas, Pedro Raminhos, Tiago, Freitas, Vasco Ramos e António Fryxell por todos os momentos de descontração e diversão.

Agradeço ao Prof. Pedro Amaral pela orientação, apoio constante ao longo da minha dissertação e pela oportunidade de fazer este trabalho de investigação. Um especial agredecimento ao Departamento de Engenharia Electroténica da Universidade Nova de Lisboa, indispensável para a minha formação académica.

Muito obrigado a todos.

Sumário:

Convencionalmente o problema do melhor caminho numa rede alude ao problema do caminho mais curto. No entanto, para a grande maioria das redes presentes nos nossos dias esta solução apresenta algumas limitações que influênciam directamente o seu bom funcionamento, bem como um aproveitamento pouco eficaz das suas potencialidades. Problemas ao nível das redes de grandes dimensões onde grafos de elevada complexidade estão comumente presentes, assim como o aparecimento de novos serviços, e suas respectivas exigências, estão intrinsecamente relacionados com a incapacidade desta solução. De modo a colmatar as necessidade presentes nestas redes, uma nova abordagem ao problema do melhor caminho deve ser explorada. Uma das soluções que mais interesse tem despertado na comunidade científica propõe a utilização de múltiplos caminhos entre dois nós da rede, onde todos eles podem passar a ser considerados como o melhor caminho entre esses nós. Desta forma, o encaminhamento deixa de ser efectuado apenas pela minimização de uma métrica, onde apenas um caminho entre os nós é escolhido, e passa a ser efectuado pela selecção de um de muitos caminhos, permitindo assim usufruir de uma maior diversidade dos caminhos presentes. O estabelecimento de multi-caminhos de encaminhamento numa determinada rede apresenta várias vantagens para a sua operacionalidade. A sua utilização pode perfeitamente melhorar a distribuição de trafégo na rede, melhorar o tempo de recuperação a falhas, ou ainda pode oferecer um maior controlo da rede por parte do seu administrador. Factores esses que apresentam ainda maior relevância quando as redes apresentam grandes dimensões, assim como quando a sua constituição é de elevada complexidade, como é o caso da Internet, onde várias redes administradas por diferentes entidades se interligam. Uma grande parte da crescente necessidade de utilização de protocolos multi-caminhos está associada ao encaminhamento efectuado com base em políticas. Pois, nesta situação, caminhos com diferentes características podem ser considerados com igual nível de preferência, e assim, fazerem parte da solução do problema do melhor caminho. Realizar encaminhamento multi-caminho usando protocolos baseados apenas no endereço de destino apresenta algumas limitações mas é possível. Conceitos de teoria de grafos e de estruturas algébricas podem ser usados para descrever a forma como as rotas são calculadas e classificadas, possibilitando a modelação do problema de encaminhamento. Esta tese estuda e analisa protocolos de encaminhamento mulito-caminho a partir da literatura conhecida e deriva uma nova condição algébrica que permite o correcto funcionamento destes protocolos sem qualquer restrição na rede. Desenvolve ainda uma série de ferramentas de Software que permite o planeamento e a respectiva verificação/validação de modelos de novos protocolos de acordo com o estudo efectuado.

Palavras Chave: Encaminhamento Multicaminho, Políticas de Encaminhamento, Problemas de Encaminhamento, Modelos Algébricos.

Abstract:

Conventionally the problem of the best path in a network refers to the shortest path problem. However, for the vast majority of networks present nowadays this solution has some limitations which directly affect their proper functioning, as well as an inefficient use of their potentialities. Problems at the level of large networks where graphs of high complexity are commonly present as well as the appearing of new services and their respective requirements, are intrinsically related to the inability of this solution. In order to overcome the needs present in these networks, a new approach to the problem of the best path must be explored. One solution that has aroused more interest in the scientific community considers the use of multiple paths between two network nodes, where they can all now be considered as the best path between those nodes. Therefore, the routing will be discontinued only by minimizing one metric, where only one path between nodes is chosen, and shall be made by the selection of one of many paths, thereby allowing the use of a greater diversity of the present paths (obviously, if the network consents).

The establishment of multi-path routing in a given network has several advantages for its operation. Its use may well improve the distribution of network traffic, improve recovery time to failure, or it can still offer a greater control of the network by its administrator. These factors still have greater relevance when networks have large dimensions, as well as when their constitution is of high complexity, such as the Internet, where multiple networks managed by different entities are interconnected. A large part of the growing need to use multipath protocols is associated to the routing made based on policies. Therefore, paths with different characteristics can be considered with equal level of preference, and thus be part of the solution for the best way problem.

To perform multi-path routing using protocols based only on the destination address has some limitations but it is possible. Concepts of graph theory of algebraic structures can be used to describe how the routes are calculated and classified, enabling to model the routing problem. This thesis studies and analyzes multi-path routing protocols from the known literature and derives a new algebraic condition which allows the correct operation of these protocols without any network restriction. It also develops a range of software tools that allows the planning and the respective verification/validation of new protocols models according to the study made.

Keywords: Multipath Routing, Routing Policies, Routing Problems, Algebraic Models.

Acronyms

AS Autonomous Systems

ISP Internet Service Provider

BGP Border Gateway Protocol

TE Traffic Engineering

IGP Interior Gateway Protocol

EGP Exterior Gateway Protocol

DV Distance Vectors

RIP Routing Information Protocol

EIGRP Enhanced Interior Gateway Routing Protocol

LSA Link State Advertisement

IS-IS Intermediate System – Intermediate System

OSPF Open Shortest Path First

MPLS Multiprotocol Label Switching

LER Label Edge Router

UCMP Unequal Cost Multipath

FEC Forward Equivalent Classes

SP Service Provider

EGPs Exterior Gateway Protocols

MED Multiple-Exit Discriminator

LOCAL_ PREF Local Preference

ECMP Equal Cost Multipath

VCs Virtual Circuits

LSPs Label Switched Paths

MIRO Multipath Inter-domain ROuting

NIRA New Internet Architecture

NRLS Name-to-Route Lookup Service

TE Traffic Engineering

Table of contents:

List of Figures

List of tables

CHAPTER 1: Introduction

Computer networks are said to be a collection of autonomous computers interconnected by a single technology, providing them capabilities to exchange data. These networks are usually connected together leading to large networks, with the Internet being the best-known example of a network of networks. Not only are they growing in size but also in complexity, where several networks are managed by different entities. The current reality in networking introduces new requirements to routing protocols, such as: policy routing; multipath routing; and traffic engineering. In fact, the unprecedented growing scale along with these new requirements leads to a routing demanding problem. Although several routing protocols have accumulated numerous extensions trying to solve this emerging problem they still face lots of challenges.

Routing can be abstracted to be a path finding in a graph and the solution to that problem have been the target of much research from different areas of the scientific community. Classical routing algorithms, like the shortest path problem, are a common form of the routing problem. Shortest path calculate the best path to a destination by minimizing some metric(s) of the links of a network graph where metrics is a property of a link: hop count; path bandwidth; path reliability; latency; and others numeric measurable costs. The best path is then the shortest path according to this metric (the path with the smallest cost). Many routing protocols are well known and use common solutions to solve the shortest paths problem. Despite the solutions found for the shortest path problem they do not solve, in general, the routing problem that we are facing in our days.

Internet is made up of a large number of independent networks or ASes (Autonomous Systems) that are operated by different organizations: usually a company; a university; or by ISPs (Internet Service Provider). Inside of its own network, within the AS, an organization can use its own algorithm for internal routing, commonly known as intra-domain routing. Typically protocols with classical algorithms, like the shortest path, are used and work properly. However between ASes, often known as inter-domain scenario, a different type of protocol must be used. Internet, for example, faces several nontechnical factors, mostly associated to business arrangements between ISPs, limiting the paths through which traffic can flow, and international boundaries, where various laws may suddenly come into play. A path selection based on a wider set of characteristics is then suitable to face inter-domain routing problems instead of a simple metric decision. In this scenario, each AS might have its own topology view, leading to different notions about what is the best path. As each AS independently chooses its own best path, the entire network routing is compromised, resulting in drastically failures. Routing loops and packets losses are frequent issues, a consequence of the ASes independent decisions.

For the inter-domain routing scenario, semantically rich concepts provide a better way to reveal the characteristics of a path. Policy based routing can facilitate the definition of paths' nature and their

relative preference, becoming “easier” to select the best path (however, and unfortunately, its implementation becomes a lot harder than for routing based on a simple metric decision). The main purpose of intra-domain protocols is to move packets as efficiently as possible from the source to the destination without having to worry about business relations (policies) between ASes, which in turn inter-domain has. Policy provide great flexibility in defining path preferences and it has been one of the most important reason for the use of BGP (Border Gateway Protocol) outside its original scope. In fact, routing based on policies is one of today’s pressure points in the routing problem.

In policy routing two different paths, according to a traditional numeric metric, can have the same policy and therefore the same preference. Paths considered with equally good preference are good to multipath based protocols but can have catastrophic results when a protocol have single path nature, like BGP (Boarder Gateway Protocol). Single nature routing protocols deals bad with multipath techniques mainly because path manipulation have to be achieve by modifying the routing state. To increase the problem single path protocols have a critical interval after a failure until the algorithm converges to a new solution. This type of problems are not verified when a multipath solution is in question, because in case of a failure, equally good alternatives paths might still be available reducing the re-converge process time, for example: using backup paths.

A protocol that plays a key role in the overall operation of the Internet is BGP which is the de facto standard protocol for the inter-domain scenario. BGP is a path vector protocol where nodes exchange routes between them. As part of the policy mechanism, routes contain path information, generally known as attributes. By analyzing those attributes, BGP can select and control paths, allowing a desired routing policy. Internet, as said, has experienced an enormous growth not only in size but also in number and variety of the currently offered applications. In other words BGP architecture faces a numerous of challenges, reflected in scalability problems and in lack of capabilities of the system. Despite the great flexibility offered by BGP, to implement different policies, policies conflicts can arise. Not only because ASes exploit these capabilities most of the times but also because of their different needs and goals. For instance, if each and every AS applies its own policy, by manipulating route attributes and filtering routes to be advertise further, it becomes extremely difficult to assure the correct operation of the protocol. These limitations, as we will see, are strictly related to the incapacity of BGP to perform routing over multiple paths. And thus, the implementation of a nature multipath routing solution seems to be one of the best possible solutions to the inter-domain routing scenario. Another issue experienced by BGP route attributes manipulation feature (leading to uncoordinated nodes), occurs when hop by hop forwarding based on the destination address is used, which may lead to an unpredicted network outcome.

A multipath solution provides more than one path solution between nodes, if available, and therefore being capable to perform some interesting services. In fact, routing based on a multipath solution, provides some huge advantages to networks management, like: performing TE (Traffic Engineering); backup paths; or load balancing. However, assuring its correct operation is not as simple as for single path routing. Scientific community is aware of the tremendous benefits that

come up with multipath and although several researches already exist, there is no present solution for the problem mentioned. The advantages, the limitations, the applicability, and the conditions for multipath routing solution success are themes explored through this dissertation.

In other words, a multipath routing solution that assures the correct behavior of multipath routing protocols is of great importance to respond the new challenges faced by today's networks. This dissertation is proposed to verify and help planning multipath routing protocols, and in order to have the more generalist solution possible, some considerations on policy based routing are required.

1.1 Motivations and Objectives

Nowadays, the majority of Internet protocols widely used, whether they are intra-domain or inter-domain, have algorithms based on single path routing solutions. For intra-domain scenarios the classical shortest path protocols are well suited. However, for inter-domain scenarios, where policies with semantically richer concepts are applied, single path routing algorithms becomes in some ways obsolete. Performing routing with rich, uncoordinated and perhaps conflicting policies is a challenge. Maintaining correctness in such protocols is exceedingly difficult and using single path routing algorithms only increases even further the complexity.

Although traffic control and multipath solution can be achieved with single path routing protocols (by updating the routing state), the amount of time and resources used are a very high cost to pay. The slow re-converge time of the algorithm and the high rate of messages to process are examples of those costs. When an AS modify its policies, all ASes having paths through it will have to rerun the routing algorithm. This will consume a great slice of time but also will trigger a great load into the network, originated by update messages. Having that in mind an assumption can be made: single path routing algorithm protocols do not suit well the need of inter-domain scenario. The unprecedented growth of networks, the increased need for backup paths and the necessity of having traffic engineering(essentially when policies come into play) led us believe that a pure multipath routing protocol can resolve many, if not all, the issues presented in these kind of networks.

Multipath routing can manage traffic control by distributing traffic through the available paths and not by changing the routing state, which is a great improvement compared to single path protocols. Multipath routing algorithms also can help with the demanding problem of policies routing in a sense that we can have two equally preferred paths without causing stress in the routing state.

A protocol based on multipath routing seems to suit well the problems faced by inter-domain networks and also for policy based routing protocols, as they are, in our time, intrinsically related. An important inter-domain routing characteristic is the independent ASes configuration and

manipulation of paths, where a minimum cooperation between ASes is desirable. This can be obtained using traditional destination hop by hop forwarding instead of alternatives methods like tunneling or source/explicit routing. The desire of having a well design protocol suited for multipath routing, having in consciousness the benefits brought by policies and by simple destination based hop-by-hop forwarding, seems to be a problem of interest.

The main goal of this dissertation is to study the characteristics and the convergence conditions that a based multipath routing protocol using simple destination based hop by hop forwarding must meet to assure its correct operation. While the study of some important aspects has been revealed, they are translated into some conditions that must be mandatorily obeyed to assure the correct operation of such protocols. The conditions presented in this dissertation provide the necessary knowledge to help planning and verifying the correct implementation of a multipath routing protocol in a network, avoiding traditional problems (such as: infinite time convergence or forwarding loops).

The approach was to use a foundation of linear algebraic structures and graph theory concepts to model the problem. The use of such models has been put to practice to model traditional shortest path protocols as well as BGP. In the concrete case of this thesis the network is modelled by directed graph and by an algebraic structure that models the calculation and ranking of paths. This dissertation uses the model described in [1] where the correct behavior of such protocols is presented.

The correct operation of a multipath routing protocols (or even in single paths routing protocols) has to be independent from the network topology. Complex networks are a field of science that models many systems in nature, providing some of the networks principles. Knowing their characteristics turns possible to generate a great variety of graphs, and thus the simulation of real networks can be achieved. This subject has a substantial interest to this thesis, which provide the capacity to test and prove the conditions for the correct operation of multipath routing protocols.

This thesis proposes to find and prove the converge conditions for the multipath routing scenario. The achievement appears to be solid and permit to solve part of the routing demand problem. A list of converge conditions that a multipath based routing protocol using simple destination based hop by hop forwarding must meet to assure its correct operation is then provided. To confirm the algebraic properties and the conditions made a software written in C/C++ language was developed.

1.2 Contributions

The main contribution of this dissertation is to provide the conditions that a multipath routing protocol must have to assure its correct operation. The properties and conditions found are based in previous works but with some improvements, particularly to assure the absence of forwarding loops in the multipath routing case. Some fundamental limits of such protocols become evident:

trade-offs between flexibility and the routing model; the amount of multiple paths that can be used simultaneously; and some more strictly restrictions that must be applied. An implementation method to calculate the routing solution using matrix iteration is also present. The outcome result is to provide a tool for help verifying and planning multipath routing protocols, with proven correct solutions.

1.3 Dissertation Organization

This dissertation is structured in five chapters and the contents of each chapter will be briefly described in this section.

Chapter 2 presents the current state of the art on Internet routing protocols and academic solutions, some pros and cons for each routing solution are analyzed. It also contains a brief discussion of how algebras appear in the routing context, and how the routing problems can be both expressed through algebraic models and solved by linear algebra with variants of classical methods. From these studies we identify important characteristics for this dissertation proposed solution. The last section of chapter 2 presents some fundamental concepts of complex topologies, which are important for the experiments realized in the fourth chapter.

Chapter 3 presents this dissertation's theoretical model for the multipath routing protocol. A set of conditions that need to be met such that these protocols operate correctly is proposed as the main contribution of this dissertation. In the proposed solution we studied the limits of providing multipath routing with semantically rich policies maintaining destination-based hop-by-hop forwarding.

Chapter 4 describes the implemented algorithms that constitutes the software and the experimented topologies. Also, by comparing the various experiment results obtained, a performance analysis of the developed software is made.

Chapter 5 presents the dissertation's conclusions and discusses of some possible future work.

CHAPTER 2: Background

This chapter introduces some major concepts related with the main subjects treated in this thesis like routing, linear algebraic concepts used in routing models and network topologies. We start with a background on routing protocols, understanding their basic operations as well as the current state of art. This knowledge will help to reveal the essential key characteristics required for the correct operation of a multipath routing protocol. We will then present the basic concepts of the theoretical models of routing (namely algebraic routing models). The final part of the chapter is devoted to network topologies and the understanding of their different characteristics.

2.1 Routing

Routing is the process of selecting the best path in a network. It is performed in many kind of networks: telephone network; data networks; and transportation networks. In this chapter, and through the rest of this thesis, our focus will be on data networks (where the Internet is included) using packet switching and destination based hop-by-hop forwarding

Packet forwarding is based on the information of routing tables, where a record of best paths (to every destinations in the network) is maintained. Routing tables are constructed using routing algorithms by computing the best paths from one node[1] (the source) to all other nodes in the network (the possible destinations). An entry in a routing table should always provide, at least, the address of the destination node as well as the link used to forward the packet.

Routing protocols are responsible for creating, maintaining and updating routing tables in nodes. To be able to calculate paths the protocol must have some information of the networks graph that can be obtained in a distributed way or via some centralized information point.

Although there are many types of routing protocols, this section presents the most used routing protocols along with their advantages and limitations. Routing protocols can be divided by their different characteristics. Firstly, routing protocols can be divided by the nature of their path calculation and forwarding algorithms in to single path routing and multipath routing. In terms of where they are used they are traditionally divided in two distinct domains, the IGP (Interior Gateway Protocols), also known as intra-domain routing, and the EGP (Exterior Gateway Protocols), or the inter-domain routing.

We will start by single path protocols dividing them according to the domains were they are used.

[1]In this thesis in general we will model networks as graphs. Using this philosophy nodes can represent network hardware devices, such as: routers, bridges, gateways, firewalls, or switches.

2.1.1 Interior gateway protocols

IGP, or intra-domain routing protocols, are protocols designed to be implemented within an AS network, controlled by a single entity (normally by network administrators). They exchange information between the nodes of a single AS and construct or update the routing tables. The main objective of intra-domain routing protocols is to move packets as efficiently as possible from the source to destination, and since all nodes are managed by the same entity they usually use simple shortest path routing,

Intra-domain shortest path routing algorithms can be divided into two major categories: distance-vector protocols and link-state protocols. A third way to perform routing inside of an AS is label switching that we will describe later in the chapter.

2.1.1.1 Distance vector routing

The term distance vector refers to protocols that manipulate vectors (arrays) of distances to other nodes in the network. Distance vector routing protocols use a shortest path routing algorithm, the Bellman-Ford algorithm, to calculate paths. Bellman-Ford routing algorithm are known to compute the routing solution by starting the calculation from the source node, which is commonly known as the right routing solution (as explained further on this chapter). The solution of this algorithm contains the smallest weight for each reachable node.

In distance vector routing protocols, nodes do not have the knowledge of the entire path to a destination. Instead they know the distance vector a vector of pairwise elements containing:

1. The neighbor that reaches the destination;

2. Distance to the destination.

Distance Vectors (DV) can contain the best path according to different metrics. A node advertises this vector through its neighbors to all the other nodes in the network and a similar advertisement is received. Using this information each node can populate its routing table. In the next advertisement cycle an updated table will be announced to the next hop of neighbors. This process continues until every routing table, of every node in the network, converges to stable values. If the protocol is operating correctly, in the end of this process each node knows the best (smallest weight) link to reach each destination.

Distance vector routing is a useful and simple technique to maintain the routing tables updated. However, in practice it has a serious drawback related to the Bellman-Ford algorithm that does not prevent routing loops. This distance vector protocols known problem is usually called the count-to-infinity problem. The reason behind this common problem is the lack of knowledge by a node of which neighbor it learns the path [2] from. This results in bad decisions (i.e. when a node goes

down). Some techniques have been developed in order to avoid the count-to-infinity problem, the split horizon with poison reverse and the hold down technique are the most usual. Although distance vector protocols converge to the correct answer, they may be slow [3]. In particularly, it reacts rapidly to good news, but leisurely to bad news (due to the count-to-infinity problems). To see how fast good news propagates consider the case: a node, initially down, comes up and the spreading rate of this information is one hop per exchange. Meaning that in a network whose longest path is of length N hops, within N exchanges everyone will know about newly revived links and nodes, for the case of bad news imagine a subnet connected like A-B-C-D-R-F, and let the metric be the number of hops. Now suppose that A goes down (from activated to deactivated). In the vector update process B notices that node A, with distance 1, is down. This happens because node B does not receive the vector update from A. The problem is, B also gets an update from C, and C is still not aware of the fact that A is down. So, C tells B that A is at a distance of two hops, which is false. This slowly propagates through the network until it reaches infinity or, if the network has N hops, only after N exchanges plus one. In which case the algorithm corrects itself, due to the relaxation property of Bellman–Ford.

Distance vector protocols are widely used in the intra-domain routing technologies in protocols such as: Routing Information Protocol (RIP) [4] and Enhanced Interior Gateway Routing Protocol (EIGRP), the most used protocol of this kind [5].

2.1.1.2 Link state routing

The count-to-infinity problem, verified in distance vector protocols affected early networks, like the APARNET [3] where it was employed. Consequently, an entirely new routing protocol, using link state routing, was implemented. The principal difference of this protocol class is related to the fact that each node maintains an overview of the entire network. In effect, the complete topology is distributed to every node by flooding the network, which overcomes the slow re-converge times but increases the network traffic.

The Link state protocol concept is that every node constructs a map of the network connectivity (in form of a graph), showing which nodes are connected to each other nodes. Since every node has the network “image” an algorithm like Dijkstra’s [6], or some of its variants, can be applied. Using Dijkstra’s algorithm each node can then, independently, calculate the best path to a destination in the network. The collection of best paths will then form the routing tables.

As said, link state routing has a relatively simple idea and is based in five pillars:

1. Discover and learn about neighbors (i.e. using hello packets);
2. Setting neighbors link costs;
3. Build link state packets, or as commonly known Link State Advertisement (LSA);
4. Distribute the link state packets to the entire network;
5. Run the Dijkstra's routing algorithm in order to create/update the routing table.

This mechanism, when compared to distance vector protocols, allows faster detection on failures and also accelerates the re-converge time to a routing solution. Although it reacts well to changes in topology it fails to scale in large networks. Mainly because each node has to compute all the LSAs from all the nodes in the network. The Link State information is flooded in the network to ensure that every nodes are updated, consuming more resources than DV [7], for example: less bandwidth will be available due to the load of LSAs traffic.

Link State routing is widely used in actual networks (within a single AS), some ISPs use the IS-IS (Intermediate System-Intermediate System) [8] link state protocol but the main link state protocol is OSPF (Open Shortest Path First) [9]. OSPF adopted from scratch many innovations designed to improve IS-IS protocol. Those innovations include: a self-stabilizing method of flooding link state updates; the concept of designated router on a LAN (Local Area Network); and a method for computing and supporting path splitting, as well as multiples metrics. Although there is a very little difference between IS-IS and OSPF, a major one can be identified. IS-IS can carry information about multiple network layer protocols at the same time, which OSPF cannot. This difference makes a huge contribution in large multiprotocol environments.

2.1.1.3 Label switching routing

In label switching protocols, the switching of network packets occurs at lower level than traditional IP routing (layer 3 in OSI model)[2]. Each packet is assigned a label number, then the switching takes place by examining the label assigned to each packet. Therefore, the switching process becomes much faster since complex lookups in routing tables, like in classical IP routing, are avoided. This kind of routing provides connection-oriented communication in which a virtual circuit is established between two endpoints before the actual data exchange begins. Technologies such as: MPLS (MultiProtocol Label Switching); ATM (Asynchronous Transfer Mode); or Frame Relay, make use of label switching.

The next section describes MPLS as it is a high performance mechanism in today's computers networks.

[2]Open Systems Interconnection (OSI) model is the reference model for networks architectures. Its layer 3, the network layer, correspond roughly to the TCP/IP (Transmission Control Protocol/Internet Protocol) model internet layer.

MultiProtocol Label Switching - MPLS

MPLS has become a widely used technology, especially in intra-domain scenarios [10]. It carries packets along a pre-determined path that is specified in advance (explicit routing), rather than routing protocols used for destination based on hop by hop forwarding. The multiprotocol feature, as its name mention, appears because MPLS headers do not belong to the network layer packet or the data link layer frame, meaning MPLS switches can forward both IP packets and non IP packets.

Figure 2.1 IP packet encapsulated by MPLS header.

Paths in MPLS are built using labels. For example, when an IP packet reaches an MPLS network entry node, commonly known as the ingress node or the LER (Label Edge Router), it inspects the destination IP address to see which MPLS path the packet should follow. Then the LER puts the right label, the correspondent label for that desired path, on the front of the packet (figure 2.1). Nodes within the MPLS network will only look to the label when forwarding the packets, avoiding exhaustive lookups in the routing tables. Since labels are short and require only exact matching, the process of finding and forwarding packets to the correct output links is extremely fast. At the other edge of the MPLS network, the egress node, removes the MPLS label and forwards the packet to the next network with its original IP header. Through MPLS networks it is possible for each flow to have its own set of labels. However, it is more common to group multiple flows that end at a particular node, or LAN, by using a single label for those paths. Flows that are aggregated together under a single label are said to belong to the same FEC (Forward Equivalent Class).

MPLS falls between the IP network layer (the layer 3 in OSI model) and the data link layer (layer 2 in OSI model). It is not a real layer 3 protocol because it depends on IP, or other network layer address, to set up the label paths. Neither is it a layer 2 protocol because it forwards packets across multiples hops. For this reason, MPLS is sometimes described as a layer 2.5 protocol of the OSI reference model.

Although the basic ideas behind MPLS are straightforward, the details are complicated with many variations and use cases. MPLS is focused for intra-domain routing and it is widely used for providing VPNs (Virtual Private Networks) over a Service Provider (SP) backbone [11].

2.1.2 Exterior gateway protocols

In contrast to the intra-domain scenario, inter-domain protocols, or EGPs (Exterior Gateway Protocols), are used to exchange routing information between independently managed ASes (representing a vital part for communications across all over the Internet). ASes are entities with different means and goals, they can be: ISPs; content providers; transit service operators; or even simple customers ASes, like a small company seeking connectivity to Internet. Unlike the intra-domain routing problem, where the same AS defines and manages the network and can set numeric metrics on links in a coherent way amongst all nodes, the inter-domain scenario brings some more challenges in maintaining the desired protocol operation. In inter-domain routing different roles influence the routing relationships between ASes, which reflect the ways routes are exchanged. Therefore, routing between ASes is not obtained by using simple metrics and choosing the path with the smallest metric, like in the intra-domain scenario. Routing decisions are made based on a set of attributes given to paths in what is commonly known as policy routing.

The de facto standard for Exterior Gateway Protocol (EGP) routing is BGP. BGP is a path-vector protocol in its fourth version and we briefly explain how it works in the following section.

2.1.2.1 Path vector protocols

A path vector protocol contains in each entry of its routing table the destination network, the next hop node and the path to reach the destination (which contains all networks crossed by the path). A great difference between a distance vector and a path vector protocol lies in this last attribute of the routing table. The presence of this attribute specifies the entire path that a packet should follow, providing a solution to the count to infinity problems verified in DV.

The solution to count-to-infinity problem can be achieved because nodes can easily verify if they already exist in the path vector, and if they do they can discard the message. Path vector based protocols have their best use in inter-domain routing scenario, particularly in the Internet. Since BGP is the path vector routing more widely used, a little description of its architecture is presented.

Boarder Gateway Protocol - BGP

BGP behaves as a path vector routing protocol, the exchanged information provides a tuple containing the paths to all reachable destinations. When an AS receives an update message containing a path to a certain destination it prepends its AS number to the path and forwards the path advertisement to valid neighbors. This, as stated previously, prevents routing loops (the count-to-infinity problem) because each message carries an ordered list of all traversed ASes. This BGP attribute is generally known as the AS level path.

Two types of sessions: i-BGP and e-BGP are present in BGP, both exchange routes between neighbors through a TCP connection. i-BGP sessions are used to exchange information about routes between the border nodes running BGP and the nodes inside the AS. The e-BGP sessions are used for export routes between domains (between ASes). BGP routes are organized in 3 tuples: destination address; AS level path and the path attributes. As it has been said, AS level path prevent routing loops by detecting if the route contains its own AS identification. Attributes are used for filtering and for decisions processes that selects the best path for each destination. The best path is then used for forwarding and it is adverted to neighbors ASes. Some attributes are presented in table 2.1.

Route Attributes	Description
Next Hop	IP address of the next-hop router along the path to the destination. On e-BGP sessions, the next-hop is set to the IP address of the border router. On i-BGP sessions, the next hop is not modified.
AS Path	Sequence of AS identifiers that the route advertisement has travel.
Local preference (LOCAL_PREF)	Attribute set by AS administrator, used to control outbound traffic. It is not attached on routes learned via e-BGP sessions, but typically assigned by the import policy of these sessions; preserved on i-BGP sessions.
Multiple-exit discriminator (MED)	Used for comparing two or more routes from the same neighboring AS. That neighbor AS can set the MED values to indicate which router it prefers to receive traffic for that destination. Inversely to LOCAL_PREF this rule tries to control inbound traffic.

Table 2.1: Important BGP attributes.

The LOCAL_PREF and the MED rules are the most interesting, they are good examples of how complex routing policies can be applied using BGP by using these attributes that precede the ASPATH in the decision process, therefore enabling the choice of paths that are not the shortest (i.e. the ones with the shortest ASPATH).

The decision process is based on the set of rules that prioritizes the importance of the route attributes, table 2.2. Once the decision process ends, the best path is installed in the forwarding table and exported to other nodes (within the same AS, using i-BGP, or to neighbors ASes, e-BGP). This routes announcement between BGP speakers[3] can be manipulated, they can be suppressed and/or aggregated. Being a vectoring protocol BGP only sends updates when changes are detected, for example: when a path in use or a policy attribute have changed. Therefore, a greater degree of scalability is available compared to other protocols, turning BGP the best suited protocol for inter-domain routing.

However the Internet large scale raises scalability and stability problems. Stability is affected by path exploration during BGP's convergence, and also by some others anomalies. It has been demonstrated [12] that during convergence, triggered by a link failure or by a withdrawal, BGP faces temporary packet loss, even if a policy compliant path to the destination might still exist. The extreme flexibility of BGP protocol in setting policies to define the best paths is the main reason for these known problems, different and independent policies applied by the ASes can lead to conflicting routing decisions and instability.

Another issue is the single path nature of the BGP protocol that limits its capabilities. All the traffic control objectives that ASes employ have to be obtained by manipulating the routing state. This produces a complex system, where some behaviors are difficult to predict [47].

Priority	Rule	Criterion
1	LOCAL_PREF	Highest LOCAL_PREF
2	ASPATH	Shortest ASPATH
3	MED	Lowest MED preferred
4	e-BGP >i-BGP	Did AS learn route via e-BGP or i-BGP?
5	IGP path	Lowest IGP path cost to next hop.
6	Router ID	Smallest router ID (IP address).

Table 2.2: How a BGP speaker selects routes.

[3]BGP speakers is the common name given to nodes running BGP.

2.2 Multipath routing

The general idea of multipath routing is to have more than one best path, between any two network nodes, in the routing table. The policy applied to a network by its administrator determines which paths can be considerate of equal preference. The list of all equally preferred paths will populate the routing table providing multiple paths.

The crescent needs of exploiting network resources, which are inefficiently used, have conducted to the exploration of multipath routing techniques. These techniques turn possible to use the bandwidth of various paths (higher bandwidths) and to improve fault tolerance, leading to a better utilization of the networks [13]. Multipath routing characteristics can also substantially improve network security mitigating man in the middle and sniffer attacks. In these attacks a particular interception point is needed. If multipath is taken into account they are harder since access to multiple paths is needed to collect the total information. Multipath can make networks more robust [3] [14] providing alternative paths in link failures without packet loss during the re-convergence time.

In summary, the most important motivations for multipath routing are:

- Better efficiency: splitting load over multiple paths;
- Better reliability: faster failover from one path to another;
- Better security: prevent on-path adversary from seeing all packets;
- More control: providing greater flexibility on paths choices.

In this section several proposals for multipath routing protocols are discussed, as well as multipath extensions for traditional single routing protocols. Similarly to the single path routing section, this section also distinguishes between the intra and inter-domain scenario.

2.2.1 Intra-domain scenario

Multipath routing have been gaining interest in network research, but in practice its use is still not widely deployed. Great part of such research has focused on the intra-domain routing scenario. Many of the works try to extend the capabilities of traditional single path based routing protocols, by introducing multipath support [15] [16] [17] using several techniques.

2.2.1.1 ECMP and UCMP

A well-known solution for intra-domain multipath routing is ECMP (Equal Cost MultiPath). The ECMP routing strategy allows packets to be forward to a single destination over multiple “shortest paths”, which are tied in terms of routing metric. Some degree of load balance can be achieved by

simple round robin packet distribution on the top of the equal cost paths. This technology can offer a better use of the network path diversity, when possible, than a normal single path routing solution [18]. However, load balancing by per-packet multipath routing is generally deprecated due to the impact of rapidly changing latency, packet reordering and maximum transmission unit (MTU) differences within a network flow, which can disrupt the operation of many Internet protocols (most notably TCP and path MTU discovery) [19]. In many situation, ECMP may not offer any real advantage over best-path routing, for example, if the multiple best next-hop paths to a destination re-converges downstream into a single low-bandwidth path (a common Internet scenario), it will merely add complexity to the traffic paths without improving available bandwidth. ECMP may also interact negatively with other routing algorithms where the physical topology of the systems differs from the logical topology, for example in systems that employ VLANs at layer 2, or virtual circuit-based architectures such ATM or MPLS.

As said ECMP is only useful when several equally “short” paths exist. Unequal Cost MultiPath (UCMP) allows the use of other routes in addition to the shortest ones. This solution has a problem, since the use of these additional routes do not guarantee loop free routing unless some loop free conditions are taken into account, which limits UCMP drastically [20].

2.2.1.2 MPLS multipath

MPLS multipath uses multipath Label Switched Paths (LSPs). We can think of LSPs as Virtual Circuits (VCs), where explicit signaling of the path in advance can establish one or more paths within the AS. In this way MPLS can establish multiple paths providing flexible splitting of traffic over the paths; flexible control over which traffic uses paths (using configurable FECs); and fast recovery from failures (by pre-configuration of backup paths) improving load balancing, traffic splitting and fault tolerance [13]. The trade-off however is that paths have to be pre-signaled complicating the configuration and operation of the network.

2.2.2 Inter-domain scenario

The main difficulties introduced in the inter-domain scenario are scalability (both at control plane and forwarding plane) and coordination amongst the independent ASes. In terms of scalability, at the control plane more paths need to be calculated and distributed, requiring more compute power process and more bandwidth for routing messages and at the forwarding level multiple best paths mean more entries in the forwarding tables.

Most of the research proposes extensions or slightly modifications to the current BGP protocol although other works come up with an entirely new multipath protocol. In the first group we have

works like: MIRO [21]; STAMP [22]; or even R-BGP [23], and in the second proposals like: NIRA [24]; Pathlet routing [25]; or Path Splicing [26].

We will briefly describe multipath BGP, MIRO, and NIRA.

2.2.2.1 BGP Multipath

As mentioned early, the current release of BGP is single path nature (only the best path is selected and advertised to neighbors). Network path diversity is not explored because possible alternative paths are not known. This poses limitations in terms of traffic engineering: possible preferred paths are not known; and traffic control can only be made at prefix level.

Multipath routing might be the solution to these BGP limitations, meaning that more routes apart from the best one have to be advertised. Some authors in [27] proposed BGP extensions to overcome the problem of not being able to advertise multiple routes. However, these multipath routing extensions can compromise scalability, since advertising more routes implies having more entries in the routing tables, as well as more traffic load in the network (due to more update messages being exchanged). Also the definition of which paths to be announced can be critical because this can enlarge the problem of inconsistent behavior, due to the independent path choices of ASes [3]. With that said improving BGP, in order to offer multipath routing, is still an open problem.

2.2.2.2 Multipath Inter-domain ROuting

Multipath Inter-domain ROuting (MIRO) is a multipath routing extension for BGP. Default paths are learnt through BGP, but additional paths can be negotiated between pairs of domains. MIRO allows an AS to request its direct neighbors for alternatives paths, to a given destination that were not considered the best one in the decision process. The request for alternative paths can be propagated through neighbors ASes, in the case that direct neighbors don't have other possible paths. A path can then be chosen and followed by establishing a tunnel between the nodes, which is identified by a tunnel ID [21].

In order to follow a specific path, tunnel ID, a packet have to be encapsulated with an extra IP header, introducing some traffic overhead. When a neighbor AS receives these packets, it then can de-encapsulate the packet and forward through a regular path to the destination, if no other alternative route exists. So, using multipath with MIRO, some parts of the traffic use destination-based forwarding while others use the tunnel ID [28]. Forwarding overhead and path negotiation cost (which introduces latency) are the prices to pay using the path selection flexibility offered by MIRO.

2.2.2.3 New Internet Routing Architecture

The New Internet Routing Architecture (NIRA) is an architecture proposal for inter-domain routing where each host has a partial graph knowledge of the AS networks to which it is connected to. This partial graph contains all connections of the AS up to the core level (ASes with no providers). With this information the host chooses routes or possible multiple routes to the destination. The system uses a mapping service called Name-to-Route Lookup Service (NRLS) to retrieve destination's addresses, since hosts only know the map to the core.

NIRA also requires use of IPv6[4] and the addressing to be hierarchical, so packets are forwarded based on the source and destination addressees in a hierarchical sense: first upwards on the user's up-graph and then downwards on the destination's up-graph.

NIRA is extremely dependent on the provider-costumer hierarchy. Although it performs well when an AS choose its own set of hierarchies, it does not suit peering links [29].

2.2.3 Summary

Multipath routing brings advantages to networks. Not only it provides a better usage of path diversity but also it provides better traffic control to networks administrators. Multipath can improve load balancing (improving throughput); reliability (less packets loss); as well as performance (less delays). Having multiple alternative paths also facilitates the application of Traffic Engineering (TE). TE is generally seen as the distribution of traffic to suit a specific goal (i.e. bandwidth requirements or congestion avoidance). With a multipath protocol this can be performed by distributing traffic along the multiple paths instead of trying to manipulate the choice of the best path so that it matches the TE goal. Although multipath presents great advantages to the routing problem, some problems arise. More routing table space is needed to accommodate the extra routes and more messages have to flow within the network announcing routes thus increasing the network load (more traffic). In summary the biggest problem that a multipath solution has to solve is scalability. Another problem is forwarding destination based hop-by-hop forwarding, which is the forwarding mechanism of the Internet. It is complicated by the possibility to forward packets along more than one path at each hop. Destination based forwarding decisions do not allow a source node to choose the entire path towards the destination and forwarding loops can occur (as we will see in chapter 3).

Despite these challenges, the advantages compared to single path routing still prevail and this is why different protocols, as the ones discuss above, try to provide efficient solutions for multipath routing.

[4]IPv6 (Internet Protocol version 6) is a newly address scheme. IPv6 is an IPv4 evolution, which provides more IP addresses.

Although there are multipath routing solutions proposed for both for inter and intra-domain scenarios, the inter-domain scenario introduces some challenging problems: ASes are independent and routing is based on policies; and alternative forwarding mechanisms are hard to implement. In summary one must have multipath routing based in policies and using simple destination based hop-by-hop forwarding. To better understand how such a protocol can be built and operated correctly, thus allowing us to build planning and verification tools, we need to model its behavior in a formal framework. In this thesis an algebraic and graph theory model is used for that effect. In the next section we present the basics needed for the reader to understand such models.

2.3 Modeling multipath routing protocols

Traditional single path routing algorithms, like shortest path, have known models and correct behavior conditions. Those conditions translate in algebraic properties exhibited by the algebraic model of the protocol. The shortest path problem, which is to find the path with the lowest cost, where cost is often a measurable numerical metric like: hop count, path bandwidth, path reliability, latency and others metrics. However, unlike a simple numeric metric, policy based routing uses policies to calculate and choose paths. A policy reflects a wider set of characteristics with semantically rich concepts, defining the nature of paths and their relative preference. Using policy based multipath routing leads to a different notion of what is the best path and the sufficient conditions for convergence are harder to define.

The concept of the best path means that preferences of the paths depend on characteristics such as business relationships with other nodes, defining in this way a hierarchical order amongst the paths. Using policy routing, two paths that are very different according to traditional numeric metrics can have the same policy characteristics and therefore have the same preference and are considered equally good. Single-path routing protocols have a critical interval after a failure until the algorithm converges to a new solution. On the contrary, in the multipath case in the event of a failure, equally good alternative paths might still be available reducing the importance of the re-convergence process. Having multiple paths also means that traffic engineering can be achieved by wise distribution of traffic amongst those paths, instead of having to fiddle with network metrics to obtain the desired result via routing state manipulations.

It seems clear that multipath routing and the need to consider it in a generalized way where paths are calculated based on policies is a promising way to perform routing. However it is also the most challenging type of routing to model and to study.

Theoretical models of routing can provide the necessary means to prove the correct behavior of routing protocols.

The most suitable theory to support routing models has been a combination of graph theory combined with algebraic concepts. Graph theory can be used to model the notion of links and nodes in the network. And algebraic concepts to model the calculation of the path weights as a composition of its links weights and the decision on which is the best weight value. The use of algebras brought interesting challenges. The study of the properties of the algebra's operators and their impact on the correct operation of the protocol is probably the most notorious challenge. Some relevant works [30] [31] [32] using algebras exist and they provide the analysis and the description of popular protocols.

As said and explained before, the simplest mechanism to forward packets, for both the intra and inter domains, is to use only the destination address. It avoids having records such as: the past route; the incoming link; or the path labels, and comprises forwarding decisions that are independent from node to node. This forwarding method scales well and therefore is very important for the adoption of multipath protocols in large scenarios like inter-domain routing. With such forwarding a special attention is needed in the study of forwarding loops.

This section will provide the theoretical algebraic knowledge needed to understand the model used in this dissertation.

2.3.1 Routing model

A routing model is said to be correct only if it complies two fundamental aspects. The first aspect of correctness is that a set of stable routes must be calculated in finite time or, in other words, it must reach a stable routing state in finite time. The second aspect is related to the absence of forwarding loops. This later aspect is particularly important if we consider simultaneously forwarding on all equally preferred paths and maintaining destination based hop-by-hop forwarding. In this thesis we consider the following definition of protocol correctness [1].

Definition 1. A routing protocol is correct if and only if:

- It converges to a routing solution, meaning it calculates a stable set of routes in finite time.

- It forwards packets without causing loops (in our case using destination based hop-by-hop forwarding).

The principal reason to model routing protocols is to understand under which conditions it can operate. In this thesis a pure algebraic setting is followed and through the use of this theoretical model we can guarantee the correctness of a multipath routing protocol. Convergence conditions are presented both for the calculations of a routing solution in finite time as well as for loop free destination based hop-by-hop forwarding on multiple paths.

Routing can be divided in two main components: policy and algorithm according to the following definition:

Definition 2. A routing protocol can be divided in two main components:

- The policy provides a way to model a meaning for the attributes of a link, a definition on how the attributes of a route are obtained from the sequence of links and the rules of preference amongst valid routes.
- The algorithm refers to the definition on how the network map is obtained; how the routes are calculated; and how routes are compared and selected according to what is defined by the policy part.

However, the evolution of routing protocols leads to a more algorithmic view turning the task of separating the policy part from the algorithm a challenge. Previous approaches to model policy rich routing protocols studied the impact of policies on individual networks, like the Stable Paths Problem (SPP) [33] framework and the Path Vector Algebras [30], [31]. Both these two approaches are theoretical models for BGP-like protocols, which limits their general application to model routing protocols. To overcome this, recent works have been done in order to define a pure algebraic setting [32], [34] that can be applied to any routing protocol.

2.3.2 Algebraic routing models

Modeling routing using algebraic structures can reveal essential aspects of a protocol, providing a means to study its properties. If a routing model uses graph theory and linear algebraic, then the separation of the two components mentioned can be achieved. Certain algebraic properties can be studied when the policy component is modelled algebraically providing a way to guarantee the correctness of a routing protocol when implemented with a certain algorithm. Graph theory is essential for routing modeling since it turns possible to model the network topology that along this dissertation will be represented by a directed graph $G(V, E)$, where V is a set of vertices and E is a set of edges.

The algebraic structure allows the modeling of three major aspects: the set of link attributes; the process of obtaining paths by appending links; and process of selecting the paths to be used.

The study of this algebraic modeling has started with traditional routing protocols, where basic algebraic structures like semirings were used [35], [37]. The use of semirings is suitable to straightforward models (i.e. shortest path) but not to model policy routing since it introduces several restrictions to models. So, a more adequate algebraic structure, with less restrictive conditions, the bisemigroup (or as we prefer, the order semigroup) can be used.

Semiring structure

The first algebraic structure to model routing protocols is the semiring [35]. A semiring is formed by a set *S*, and its elements model link and path attributes. Normally, there are two special elements: $\bar{0}$ which usually models a non-existent or invalid path; and $\bar{1}$ to model the trivial path, or in other words, the path from one node to itself containing no links. The set *S* is endowed with two binary operations $\oplus$ and $\otimes$ with the following properties:

- $\oplus$ is commutative: $\forall\ a, b \in S\ a \oplus b = b \oplus a$;

- $\oplus$ also should be idempotent: $\forall\ a \in S\ a \oplus a = a$;

- $\otimes$ distributes over $\oplus$: $\forall\ a, b, c \in S\ a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$;

- $\oplus$ has identity $\bar{0}$ which is an annihilator for $\otimes$;

- $\otimes$ has an identity $\bar{1}$.

The $\oplus$ operator models the path choice, and that is why it should be idempotent. This operator performs the selection of the best path accordingly to the network policy, which can be a rich policy or a simply metric minimization. The $\otimes$ operation models the paths calculation, it states the result when two links are linked together, for example: in shortest path routing it represents the sum of the metrics of the links. The same is to say the $\otimes$ operation is applied between a link value $l \in S$ and a path value $p \in S$ resulting in another path value, which has to belong also to the set S. This operator expresses how the attributes of a path change with the addition of links.

The property of distributivity of $\otimes$ over $\oplus$ present on semirings holds in traditional routing models like the shortest path. Mainly due to the straightforward characteristic of such structures. But for link the weight abstraction, presented in policy routing, this characteristic of semirings is too restrictive, limiting the capabilities of the operators [35]. So, a more general algebraic structure is needed to model multipath routing protocols with policies. An algebraic structure similar to a semiring but without distributivity is called a bisemigroup. If instead of a second binary operation we consider and order we have an order semigroup.

Order semigroup structure

A semiring structure does not suit well to model policy based routing protocols. Since this dissertation tries to give the more general concepts possible to model any type of protocol, we cannot use semiring as our modeling structure. Semiring are proper to model metric based shortest path routing protocols but, in order to use a more generalist model we have to use an algebraic structure that does not need any properties to be present.

Definition 3. A bisemigroup is an algebraic structure formed by a set S endowed with two associative binary operations $\oplus$ and $\otimes$ such that we have:

Associativity of $\oplus$: $\forall\ a, b \in S\ a \oplus (b \oplus c) = (a \oplus b) \oplus c$

Associativity of $\otimes$: $\forall\ a, b \in S\ a \otimes (b \otimes c) = (a \otimes b) \otimes c$

Although bisemigroup structures can be used to fulfill our desire, the order version called order semigroup [32] is, from our point of view, a more perceptible algebraic model. For instance, if $\oplus$ is idempotent[5] we can define a canonical order $\preccurlyeq$ induced by $\oplus$ as:

$$b \preccurlyeq a \text{ if } a \oplus b = b$$

The application of this operator can model path selection preference in the set S, meaning that for all $a \in S\ \bar{1} \preccurlyeq a \preccurlyeq \bar{0}$. Using such operator the set order is bounded between the trivial path (which is the more preferred) and the invalid path (the last preferred). So, if we substitute the $\oplus$ operator by the derived $\preccurlyeq$ order we have the order semigroup.

Definition 4. An order semigroup is an algebraic structure such that:

$(S, \preccurlyeq, \otimes)$

The use of bisemigroup or order semigroup is merely a question of considering the path selection as the binary operation over two paths weights or as the result of the relative position of the paths in the order $\preccurlyeq$.

2.3.3 The routing problem

The network is modeled by a directed graph $G = (V, E)$, where V is a set of vertices and E is a set of edges. If we consider a weight/attribute function $w(i, j)$ that maps $E \rightarrow S$ (providing values to edges), a path is a sequence of edges connecting vertices, $P = v_1, v_2, \ldots, v_k$ and its weight is given by:

$$w(P) = w(v_1, v_2) \otimes w(v_2, v_3) \otimes \ldots \otimes w(v_{k-1}, v_k)$$

The routing problem is to calculate for all pairs $(i, j) \in V$ the weight of $P(i, j)$ which defines the set of all paths between i and j, and by using $\oplus$ the set $O(i, j)$ of the smallest weight path is obtained (the set of the best paths):

[5]Idempotent is a property of certain operators. They can be applied multiple times without changing the result beyond the initial application.

$$O(i, j) = \bigoplus_{p \in P(i,j)} w(p)$$

The solution can be obtained by taking the matrix version of operations $\oplus$ and $\otimes$, solving one of the two equations: the left equation (2.1); or the right equation (2.2) [37] [38]:

$$L = (A \otimes L) \oplus I \qquad \text{(Eq. 2.1)}$$

$$R = (R \otimes A) \oplus I \qquad \text{(Eq. 2.2)}$$

Where I is the identity matrix and A is the adjacency[6] matrix of the network graph:

$$A(i, j) = \begin{cases} w(i,j), & \text{if } (i,j) \in E \\ 0, & \text{otherwise} \end{cases} \qquad \text{(Eq. 2.3)}$$

R and L are the matrices with the smallest weight of the paths values between nodes i and j, given the set of the smallest weights paths of the neighbors q of source i.

$$L(i, j) = \bigoplus_{q \in V} A(i, q) \otimes L(q, j) \qquad \text{(Eq. 2.4)}$$

$$R(i, j) = \bigoplus_{q \in V} R(i, q) \otimes A(q, j) \qquad \text{(Eq. 2.5)}$$

The difference between the left and the right equation resides in the order of the path weight calculation. The left equation (2.4) starts from the destination whereas the right equation (2.5) starts from the source. In both equations, the solution contains the smallest weight from source to destination and not actual paths. Although, it is possible to store the nodes that correspond to these weights during the calculation process and therefore obtain the routing table. The solution for those equations can be found by calculating the closure[7] matrix of the adjacency matrix A (denoted by A^*) through matrix iteration. During matrix iteration it is feasible to consecutively choose the more preferred path (accordingly to $\preccurlyeq$) between each (i, j) pairs of G.

These method of matrix calculation are related to the Bellman-Ford and the Dijkstra's algorithm which are well known to compute their solutions [39]. However, as we will see in the next section, they provide different solutions to different conditions of the A^* computation.

2.3.4 Convergence conditions

The computation of A^* can be assured by two sufficient, but not necessary, conditions: monotonicity and the increasing property.

[6]The adjacency matrix is a mean of representing which vertices of a graph are adjacent to which other vertices.

[7]A closure matrix, known as A^*, contains the final solution of matrix multiplication. When using one of the algorithm described it represents the best weight between every vertices in a graph.

Definition 5. An order semigroup (S, $\preccurlyeq$, $\otimes$) is monotone if:

$$\forall\ a, b, c \in S\ a \preccurlyeq b \Rightarrow (c \otimes a) \preccurlyeq (c \otimes b)$$

This condition corresponds to distributivity of $\otimes$ over $\oplus$ in a bisemigroup. A bisemigroup has distributivity of $\otimes$ over $\oplus$ if:

$$\forall\ a, b, c \in S\ c \otimes (a \oplus b) = (c \otimes a) \oplus (c \otimes b)$$

This first condition is monotonicity for order semigroup, or the distributivity of $\otimes$ over $\oplus$ in a bisemigroup. The best path is the path that is lowest in the $\preccurlyeq$ order. Intuitively, in a monotone order semigroup a sub path of a best path is also a best path. This is a sufficient condition for the existence of A^*. In this case a global optimum solution for the routing problem is obtained, either by solving the left or the right equations.

Definition 6. An order semigroup is increasing if:

$$\forall\ a, b \in S\ a \prec b \otimes a$$

The second condition is the increasing property for order semigroup. This means that any addition of a link to a path decreases its preference, increasing its position in the order $\preccurlyeq$. If an order semigroup is not monotone but only increasing we do not assure a global optimum solution because it is not possible to guarantee that every sub path of a best path is also a best path. However a local optimum solution for the left and for the right equations can still exist. They might not be equal because the best routes in one node are dependent on the best routes of the other nodes [40].

Although the two above conditions are enough to provide a routing solution for A^*, the network's graph must comply with the non-negative cycle condition [35]. Where a cyclic path C is a path where all nodes are different apart from the first and last nodes which are the same.

> Condition 1. Let's denote w(p) as the value in S that corresponds to the result of the path composition operation $\otimes$ over the links l_i of p. The network meets the non-negative cycle condition if and only if $\bar{1} \preccurlyeq w(C)\ \forall\ C \in G(V, E)$.

The condition above is naturally met for every network labeled with a set S that has as lower bound its trivial path $\bar{1}$. If the condition is not met, travelling around the cycle will continuously produce a more $\preccurlyeq$ preferred path, and the algorithm does not stop since in each iteration the preference is better.

The non-negative cycle is naturally met for an increasing order semigroup since every link added to the path decreases its preference (meaning that travel the cycle also decreases the path preference). However, the non-negative cycle can become a problem when a monotone order

semigroup is in use. To prevent this possible problem a stronger condition must be set on all entries of the adjacency matrix A^*:

$$\bar{1} \preccurlyeq A(i, j) \ \forall \ i, j \in E$$

In this last case the previous non-negative cycle condition will always be met, this is because all edges have a weight equal or worse than $\bar{1}$, which means the preference will not increase in a monotone order semigroup with the addition of any link.

At this point the necessarily algebraic properties for the convergence solution of a routing protocol have been established. However, and since this dissertation is focused in destination based hop by hop forwarding, it is important to make some considerations on the algorithm to use.

Destination based hop by hop convergence conditions

As described in definition 1, a routing protocol is only correct if it converges in a finite time to a routing solution and if it forward packets without causing loops even if using all equally preferred paths at the same time. So, the algorithm part must assure that no packets are looping within the network. Must of the times forwarding loops are caused by independent forwarding decisions that are not consistent, which is the case of destination based hop by hop forwarding.

In a strictly monotone algebra the sub paths of the best path are always the optimal path and therefore decisions of the next hops will be consistent (global optimum solution). If monotonicity is not present then a forwarding loop can occur. A local optimum solution which depends on the preferred paths of neighbors, can then be a solution. However a distinction between the left (2.1) and right (2.2) equation must be made. It is well know that right local solutions can cause forwarding loops in destination based hop by hop forwarding [39]. The reason is that equation (2.2) path weights are calculated from source to destination, and since the best path for the next hop can be different, a forwarding loop can occur. In equation (2.1) the calculation is done from destination to source and even in the absence of monotony, forwarding loops do not occur since path weights are always dependent on the weights of the sub-paths from the next hops to the destination.

Convergence impact of relaxing or forcing strictness in $\otimes$

The sufficient conditions for convergence to a routing solution are: the non-strict version of monotony and/or the strict increasing property. Now, we will see if convergence is still possible, and in which conditions, when changing these two conditions between their strict and non-strict version.

Definition 7. An order semigroup $(S, \prec, \otimes)$ is strictly monotone if:

$$\forall\ a, b, c \in S\ a \prec b \Rightarrow (c \otimes a) \prec (c \otimes b)$$

The non-strictly monotone algebra condition, seen on definition 5, assures convergence to a routing solution if the network meets the non-negative cycle condition (condition 1). In such algebras a path travelling around a cycle never decreases in value and therefore it is always increasing or maintaining its position in the $\preccurlyeq$ order. However, when a strictly monotone algebraic condition, like the one presented in definition 6, is present the following statement might be true:

$$\exists_{a,b \in S} \text{ such that } a \preccurlyeq b \text{ and } (c \otimes a) \simeq (c \otimes b)$$

Meaning that the preference can be maintained with the addition of links to a path. Consecutively the network can have an infinite number of paths with the minimal weight which can lead to non-convergence to the routing solution. Although it is possible to converge to the minimal global optimum values (because they never increase), it is not possible to obtain them in finite time. Which is not a very good practical case when implementing a protocol with such an algebra.

Definition 8. An order semigroup is non-decreasing if:

$$\forall\ a, b \in S\ a \preccurlyeq b \otimes a$$

The strictly increasing order semigroup condition, definition 6, assures convergence to a routing solution. In this case adding a link to a path always increases the path weight and therefore decreases preference. This means that preferences around cycles always decreases, and we always converge to a routing solution in finite time.

Convergence conditions summary

The study made so far permits to conclude that, in order to assure convergence to the routing solution, we need a protocol modeled by an algebraic structure of one of the following two types:

- A strictly monotone order semigroup according to definition 7.
- An order semigroup with a strictly increasing $\otimes$ operation according to definition 6.

This means that only with one of the two strict versions of the properties we can assure finite time convergence, and therefore guarantee the correct operation of a routing protocol policy part. The forwarding correctness is a different problem that we address later.

Algebraic structures with the above properties are quite limiting in the number of equally preferred paths that we might have since it tends to consider paths of the same preference when they have

the same number of links. So, in the next section we present a model for multipath routing so that we can study if correct behavior can be obtained with less strict conditions.

2.3.5 Modeling multipath

Multipath differs from single path in two aspects. Firstly all equally preferred paths are part of the routing solution and therefore the non-converge problem can be worse. Secondly it implies the use of more than one path to forward packets simultaneously to the destination and this complicates the assurance of correct forwarding behavior. Therefore, it is important to derive the convergence conditions in the multipath case, and verify if they are similar to the ones of the single path routing problem.

The multipath routing case can be modeled by defining minimal sets of the set S [1].

Definition 9. A minimal set is a subset $C \subset S$ with cardinality $|C| = n$ such that:

$$C = \min^{\preccurlyeq}(C) = \{x \in C \mid \forall\, y \in C: x \preccurlyeq y\}$$

Minimal sets model the multipath problem. The multipath case implies that $\oplus$ might not be selective, $\forall_{a,b \in S}\ a \oplus b = a \cup b$, if we consider paths with different weights to be of equal preference and in the order $\preccurlyeq$ we can have $a, b \in S$ with $a \neq b$ but $a \simeq b$, meaning that a and b have an equivalent position in the order (which is a good characteristic in order to model policy based routing).

A minimal set represents a set of paths with the same preference, which means that they are either equal $w_1 = \ldots = w_n$ or equivalent $w_1 \simeq \ldots \simeq w_n$. The set of all minimal sets of S is $M(S)$.

An algebra can be defined from an order semigroup $(S, \prec, \otimes)$ in which the minimal sets of S are used and the operators are redefined to work with the minimal sets instead of the individual elements of S. The operator $\oplus'$ is given by:

$$A \oplus' B = \min^{\preccurlyeq}(A \cup B)$$

And returns the most preferred paths out of the union of A and B. Likewise, $\otimes'$ is the extension of $\otimes$ to sets of paths:

$$\forall_{a \in S, C \in M(S)}\ a \otimes' C = \min^{\preccurlyeq}(a \otimes C)$$

Meaning that $\otimes'$ results in the minimal path weights when $\otimes$ adding a new link a to all elements of the set of minimal paths C for all existent minimal set $M(S)$. The resulting multipath algebra is then:

$$(M(S), \preccurlyeq', \otimes')$$

If an algebra is monotone the minimal set routing solution can be found and it is a global optimum solution [41]. In the absence of monotony, an algebra with a strictly increasing $\otimes$ operation is sufficient to assure convergence to a local optimum solution using multiple paths of equal or equivalent preference. In [42] the author uses a policy relation to study the impact of multipath in the policy part of routing protocols. The conclusion is that a multiple path routing solution is possible if the policy relation is anti-reflexive[8] (an anti-reflexive policy relation can be mapped into an increasing algebra in a pure algebraic setting).

In summary, the convergence conditions for multipath are the same as in the single path case. The policy part of the protocol has either to be a monotone order semigroup (definition 5) or an order semigroup with an increasing $\otimes$ operation (definition 6). However, this conditions are too strict to model a multipath routing protocol, as we now examine.

Convergence conditions considerations

In this thesis we want to study multipath routing models with capabilities to model multipath routing protocols in the most possible flexible way possible (protocols with policy-based routing for example). These algebraic properties impose too much restrictions to a multipath routing protocol. So, before further explanations let us see what this means in terms of protocols that can be modeled by structures with such properties.

One algebraic property that assures convergence is strict monotony. However, it is well known that monotony is hard to maintain in policy routing protocols, mainly due to the existence of invalid paths [30] [39]. A good example are the common policies used in the inter-domain scenario (i.e. Internet routing between ASes). Consider for example the set:

$$S = \{p2c, c2p, p2p\}, \text{ where: } \begin{cases} p2c \text{ is a provider to customer link} \\ c2p \text{ is a customer to provider link} \\ p2p \text{ is a link between peer nodes} \end{cases}$$

A common policy in the Internet is to prevent paths learned from providers to be advertised to other provider ($p2c \otimes c2p = \overline{0}$), another common policy is to prefer customer routes to provider routes ($p2c \prec c2p$). This policies, when together, lead to monotony loss:

$$p2c \prec c2p \text{ but } c2p \otimes c2p = c2p \prec p2c \otimes c2p = \overline{0}$$

This example illustrates how monotony is hard to maintain in policy routing. In other words, and using a more generic and simple example, consider for a given protocol:

[8]Where a reflexive $\preccurlyeq$ operator: $\forall\ a \in S\ a \preccurlyeq a = a$. Which means that an anti-reflexive relation always meet: $\forall a, b \in S\ a \preccurlyeq b = a$ and not $a \preccurlyeq b = b$.

$$a \preccurlyeq b \preccurlyeq c \preccurlyeq \overline{0}$$

that for policy reasons adding a link with label x to a path with weight a is forbidden,

$$x \otimes a = \overline{0}, \text{ but } x \otimes b = c \Rightarrow \text{monotonicity is broken since } c \preccurlyeq \overline{0}$$

Related work has produced results for order semigroups that hold some of these properties.

The other algebraic property that still provides convergence assurance is the increasing property. But, in order to have a strictly increasing $\otimes$ operation, the path weight has to strictly increase with any added link. This means that it will be difficult for two paths with a different number of hops to have the same preference in $\preccurlyeq$. This is a limitation in the use of the possible path diversity in a network, which is not ideal for a multipath routing model.

Since the necessary convergence conditions, described so far are not suited to model a multipath routing protocol, it is of interest to have a $\otimes$ operation that can be merely non-decreasing as defined in definition 8. This provides the flexibility to allow more paths with a different number of hops to have the same preference, and therefore increase the amount of paths that can be used simultaneously to forwarding. Note also, that monotonicity implies a non-decreasing $\otimes$ operation and strict monotonicity implies that $\otimes$ is increasing (although the inverse is not true).

The problem to model a protocol using an order semigroup with a non-decreasing $\otimes$ operation is that such algebra does not guarantee convergence to the routing solution (as defined in definition 1). In this case the $\otimes$ operation is not strictly increasing, meaning that cycles can be present in the network because the path weight preference is maintained with the addition of consecutive links around a cycle. Once the algebra by itself does not assure convergence, some extra restriction on the network have to be set (assuring that problematic cycle are not present in the network).

Convergence in non-decreasing models

For single path routing forwarding loops do not occur in destination-based hop-by-hop forwarding if the solution is calculated by applying $\otimes$ to the left. Multipath routing introduces a new situation in forwarding that needs to be taken in consideration when evaluating if a cycle causes problems.

The presence of cycles in a network with a non-decreasing $\otimes$ operation can be formalized as a generalization of the non-negative cycle condition, condition 1. An algebraic generalization of the non-negative cycle is found in [30], and the author called it a free cycle. A re-definition of the notion of a free cycle in a pure algebraic setting for an order semigroup is made [1].

Definition 10. Free cycle. Consider a cycle *P*= v_1, v_2, …, v_{k-1}, v_k = v_1. Consider d a destination, V_i the set of all paths from vertex v_i to d, and $\alpha_{ij} \in S\backslash\overline{0}$[9] the weight of the *j* path starting at vertex v_i to destination *d* without going through v_{i+1}, and therefore, $\alpha_{ij} \in A_i$. Consider the sets of all paths to d from every vertex of the cycle, $V_1 \cup V_2 \cup \ldots \cup V_{(k-1)} \cup (V_k = V_1)$. The cycle is free if there is at least a path *j* starting at node *i* ($1 \leqslant I \leqslant k - 1$) such that $\alpha_{ij} \prec w(v_i, v_{i+1} \otimes \alpha_{(i+1)k})$, $\forall_{k \in V_{i+1}}$. This has to be valid for all reachable destinations, *d*.

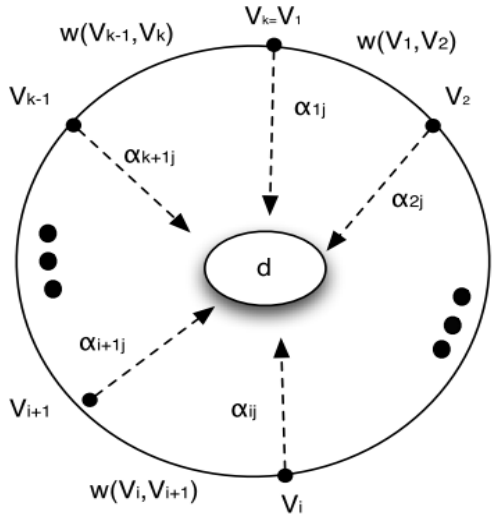


Figure 2.2: Free cycle illustration [1].

In other words, a cycle to be free must at least have one of the dotted line paths α_{ij} to *d* as a more preferred path at node v_i than the path in solid lines to $v_{i+1} \otimes$ added to the next $\alpha_{i+1,j}$ path (as figure 2.2 illustrates).

For example: consider that in the cycle of figure 2.2 all links between V_{k-1} and V_3 are labelled with *b* in both directions forming equal preferred sub-paths, $w(V_{k-1}, V_k) = w(V_k, V_{k-1}) = \ldots = w(V_2, V_3) = w(V_3, V_2) = b$. Consider V_i (which does not belong to the sub-path) and the links from V_{k-1} to V_i and from V_3 to V_i have weight *w* = *a* in the forward direction and *w* = *c* in the reverse direction. If α_{ij} is preferred among every other dotted lines in the cycle than finite time convergence is assured because paths around the cycle do not maintain preference.

This means that it always exists at least one path leaving the cycle that is more preferred than the ones across the cycle. This conclusion is in concordance with the convergence conditions (seen previously) for the strictly version of monotonicity and increasing algebras, as long as the network graph is labeled in a set S with a lower bound $\overline{1}$ (non-negative cycle condition).

[9] $S\backslash\overline{0}$ means that the non existence path will not be taken into account.

However, in multipath with a non-decreasing operation, forwarding loops can still occur inside a sub-path that maintains preference. Consider (figure 2.2) that the node V_k = V_1 has two equally preferred paths to V_i (clockwise and anti-clockwise). If V_2 also has two equally paths to V_i than a forwarding loop can occur between $V_1$and V_2. Packets bouncing between V_1 and V_2 may never reach the destination *d* which is problematic, the next chapter provides a solution to this kind of problem that arises from the possibility to forward packets on all equally preferred paths something that is not considered in most works.

CHAPTER 3: Multipath routing using destination based hop-by-hop forwarding

3.1 Introduction

Current results in modeling multipath routing protocols only prove correct behavior for models that exhibit a strict decrease in the preference of a path with every added link (the strictly increasing $\otimes$ operator). Which is a very strict condition for a multipath routing protocol since it does not allow equivalent paths with similar policy but a different number of links. So, in order to model the multipath case a relaxation of the $\otimes$ operator to a non-decreasing operation is a must. The solution, provided in the previous chapter (the free cycle condition, definition 10) is to apply some restrictions in the network graph, thus assuring that problematic cycles are non-existent. Although this solution works fine for the single path case, in multipath, if forwarding is restricted to only one of the equally preferred paths, it does not suit well the multipath forwarding case, where all equally preferred paths can be used simultaneously.

The incorrect behavior in this case is primarily caused by graph circuits where the policy part of the routing protocol is incorrectly applied (definition 2). The concept of a directed cycle is introduced to study the correctness conditions for multipath forwarding and conditions for correct operation without the strict increase operator are obtained. These conditions can be used at a multipath protocol design stage (to define the policy set and which policies can be applied in a circuit) as well as to assure the correct operation of a multipath protocol (by verifying its properties).

In this chapter, the least restrictive sufficient conditions for the correct operation of a multipath routing protocol using destination based hop by hop forwarding are explained (meaning that a stable routing solution is obtained in finite time and no forwarding loops occur). The first section re-defines some notions of section 2.3 and contextualizes the reader to the routing model used through the rest of this dissertation.

3.2 Multipath routing model

In this dissertation, the approach used to model a multipath routing protocol is a combination of graph theory with algebra. The network model (which model the notions of link and node) is represented by a direct graph $G(V,E)$, where V is a set of vertices and E is a set of edges. The set of link attributes, the process of obtaining paths by appending links and the process of selecting the paths to be used are modeled by an algebraic structure. The algebraic structure used is an order semigroup structure, formed by a set S endowed with a binary operation $\otimes$ and an order $\preccurlyeq$:

$$(S, \preccurlyeq, \otimes)$$

The elements of S represent link and path attributes, $\overline{0}$ models a non-existent or an invalid path, and $\overline{1}$ the trivial path.

The $\preccurlyeq$ order models the path choice and should be reflexive:

$$\forall\, a \in S,\ a \preccurlyeq a = a;$$

and transitive:

$$\forall\, a, b \in S,\ a \preccurlyeq b \wedge b \preccurlyeq c \Rightarrow a \preccurlyeq c.$$

since we want to have the possibility of having equivalent or incomparable paths,

Antisymmetry:

$$\forall\, a, b \in S,\ a \preccurlyeq b \wedge b \preccurlyeq a \Rightarrow a = b;$$

and totality:

$$\forall a, b \in S,\ a \preccurlyeq b \vee b \preccurlyeq a;$$

are not present, meaning that $\preccurlyeq$ is a preorder.

The $\otimes$ operation models the calculation of paths and a path is a sequence of edges connecting vertices, $P = v_1, v_2, \ldots, v_k$. Its weight is given by $w(P) = w(v_1, v_2) \otimes w(v_2, v_3) \otimes \ldots \otimes w(v_{k-1}, v_k)$. For all pairs (i, j) with $i, j \in V$ the weights of all paths between i and j are calculated and therefore the set $O(i, j)$, containing the smallest weight paths, is obtained using the preorder $\preccurlyeq$. The result is the routing solution.

Since antisymmetry and totality are not present, the concept of minimal sets in S can be applied to model a multipath routing protocol. The order semigroup $(S, \preccurlyeq, \otimes)$ is then capable to be redefined as $(M(S), \preccurlyeq', \otimes')$, in order to work with the minimal sets instead of the individual elements of S.

Using such model we can study and analyze multipath protocols correctness, for both statements of definition 1 in abstract terms. See chapter 2, in special section 2.3, for details.

3.3 Convergence to a fixed routing solution

As stated in section 2.3.3, the routing solution can be obtained via matrix iteration:

$$\sigma: L \rightarrow (A \otimes L) \overset{\preccurlyeq}{}_{min} I \qquad \text{(Eq. 3.1)}$$

where $\overset{\preccurlyeq}{}_{min}$ is the minimization in the order $\preccurlyeq$, I is the identity matrix and A the adjacency matrix representing the directed graph. At the k_{th} iteration $\sigma_{i,j}^{k}$contains the set of best paths (according to the preorder):

$$\sigma^k = A^k \overset{\preccurlyeq}{}_{min} A^{k-1} \overset{\preccurlyeq}{}_{min} \dots \overset{\preccurlyeq}{}_{min} A^2 \overset{\preccurlyeq}{}_{min} A \overset{\preccurlyeq}{}_{min} I \quad \text{(Eq. 3.2)}$$

If the iteration converges to a stable routing solution, the limit matrix for $k \rightarrow \infty$ is $\sigma^k = A^*$, where A^* is the closure matrix of A. A^* contains in each position the set of best paths amongst all paths of any length. Several proofs [30] [32] [34] can be found in literature that for a non-monotone order semigroup with an increasing $\otimes$ operation the matrix σ converges, in this case to a local optimum solution. It is well-known that to obtain loop free forwarding (choosing a single path from the equally preferred set) in these models, paths must be calculated from destination to source, or the same is to say: using the left local solution (which can be implemented with the Dijkstra's algorithm, as stated in 2.3.3).

The multipath case

The solution σ contains all the paths between (i, j) belonging to the minimal set according to $\preccurlyeq'$, corresponding to the multipath routing solution. The convergence of σ in this case is strictly related to the presence of circuits in the network. Those circuits can be described as:

> **Definition 13**. A closed circuit, or circuit (for simplicity), C is a sequence of connected vertices such that the first and the last are the same:
>
> $C = \{v_0, v_1, v_2, \dots, v_k = v_0\}$

Since the protocols that we are modeling have directed edges, we can define paths circling the entire circuit in each direction, in what we called directed cycles:

> **Definition 14**. Consider a node p in a circuit. A directed cycle is a path from the node p to itself around the circuit in one direction:
>
> - C_p^R, if it is in the clockwise direction (right);
> - C_p^L, if it is in the counter-clockwise direction (left).

Definition 15. If the directed cycle does not cross the entire circuit, but only a segment until a node $l \neq p$, $C^R_{p,l}$and $C^L_{p,l}$ denotations can be followed for the respective right and left directions of the cycle segments.

The next figure, figure 3.1, illustrates a closed circuit (formed by k – 1 vertices/nodes) as defined above. A destination node d is also illustrated and might be reached (directly or not) by any of the nodes presented (V_1 to V_{k-1}). Following definition 14, the weight of the directed cycle, in the right direction, of node V_1 is:

$$w(C^R_{V_1}) = w(V_1, V_2) \otimes \ldots \otimes w(V_i, V_{i+1}) \otimes \ldots \otimes w(V_{k-1}, V_1);$$

And following definition 15, the weight of a cycle segment from V_{k-1} to V_2 can be expressed as:

$$w(C^R_{V_{k-1},V_2}) = w(V_{k-1}, V_1) \otimes w(V_1, V_2).$$

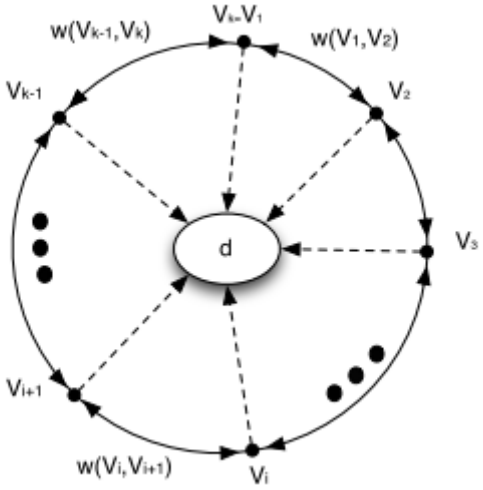


Figure 3.1: Generic cycle.

Consider a node V_d (more than one might exist) belonging to the circuit and that is connected to destination *d* by a path that exists the circuit (i.e., through paths illustrated as dotted lines in figure 3.1). This means the other V_k vertices in the circuit have two possible paths for each V_d, by using the left and right directions (definition 15). If we consider that both path weights belong to the minimal set of the most preferred weights according to $\preccurlyeq'$:

$$w(V_i, d) = \begin{cases} w(C^R_{V_i,V_d}) \otimes w(V_d, d) \\ w(C^L_{V_i,V_d}) \otimes w(V_d, d) \end{cases}$$

Now, supposing that these paths are extended through all links of the circuit back to V_i ($w(C^R_{V_i}) \otimes w(C^R_{V_i,V_d}) \otimes w(V_d, d)$ and $w(C^L_{V_i}) \otimes w(C^L_{V_i,V_d}) \otimes w(V_d, d)$), which results in a path also belonging to the minimal set. The number of paths with weights belonging to the minimal set is therefore

infinite since consecutively ⊗ adding $w(C_{V_i}^R)$ or $w(C_{V_i}^L)$ results in new paths belonging to the minimal set.

So (and as stated in chapter 2 for monotonicity), a protocol modeled by a non-strict monotone ⊗ operator can assure convergence to the minimal set weights but does not obtain all the paths belonging to the set in finite time. To prevent this situation, and without adding complexity to the solving algorithm to exclude cyclic paths, the following condition must be met:

> Condition 2. Consider a circuit $\{V_1, V_2, V_2, \dots, V_k = V_0\}$. For all destinations *d*, and all $V_d's$ vertices (as defined early), and all vertices V_i in the circuit:
>
> $$w(C_{V_i,V_d}^R) \otimes w(V_d,d) \prec w(C_{V_i}^R) \otimes w(C_{V_i,V_d}^R) \otimes w(V_d,d)$$
>
> $$\wedge$$
>
> $$w(C_{V_i,V_d}^L) \otimes w(V_d,d) \prec w(C_{V_i}^L) \otimes w(C_{V_i,V_d}^L) \otimes w(V_d,d)$$
>
> $$\wedge$$
>
> $$w(C_{V_i,V_d}^R) \otimes w(V_d,d) \prec w(C_{V_i}^L) \otimes w(C_{V_i,V_d}^R) \otimes w(V_d,d)$$
>
> $$\wedge$$
>
> $$w(C_{V_i,V_d}^L) \otimes w(V_d,d) \prec w(C_{V_i}^R) \otimes w(C_{V_i,V_d}^L) \otimes w(V_d,d)$$

Finite time convergence

It is easy to see that both with a strict increase operation and/or strict monotonicity an order semigroup, naturally meets condition 2 independently of the number of cycles present in the network's graph.

However, for the multipath case, we have seen that these properties restrict the amount of paths of equal weight and therefore the number of usable paths. By relaxing the ⊗ operator to a non-decreasing property, the importance of the number of hops is reduced and the possibilities to have more paths of equal preference increase. In this case, condition 2 might be broken because some circuits of the network can exist where:

$$w(C_{V_i,V_d}^X) \otimes w(V_d,d) \simeq w(C_{V_i}^X) \otimes w(C_{V_i,V_d}^X) \otimes w(V_d,d) \text{ with } X \in \{R, L\}$$

From condition 2 (and since $a \succcurlyeq a \otimes b$ cannot occur), it is not possible to assure finite time convergence with a non-decreasing property since the set of all paths with minimal weight might not be obtained in finite time due to a possible infinite number of equal preferred paths by just

travelling around a circuit. This means that the network must not contain any circuit *C* with edges *E* labeled in *S* such that a $V_i \in C$ exists where either $u \simeq C_{V_i}^R \otimes u$ or $u \simeq C_{V_i}^L \otimes u$ for all $u \in S$. If no such circuit exists then condition 2 is met and therefore the network operates correctly.

In summary convergence to the multipath routing solution in finite time can be obtained in a non-decreasing $\otimes$ if:

- σ is calculated using an algorithm that stops the iteration when travelling around circuits;
- or by restricting the network so that there are no circuits in the graph where the weight of its links can maintain the preference of a path.

If we do not wish to change the path calculation algorithm to detect cycles then a condition for circuit labeling must be followed.

> Condition 3. Consider a set $L_u \subset S$ for each value $u \in S \backslash \bar{0}$ with the following characteristic: $L_u = \{l \in S \mid \text{such that } u \simeq l \otimes u\}$. This means that for each value in *S*, the corresponding L_u set contains the values of *S* that result in an equivalent preference with the $\otimes$ addition. A circuit cannot have all its links *E* labeled with values *l* belonging to the same L_u set neither in the *R* nor in the *L* directions.

For a non-decreasing $\otimes$ operation and if the condition above holds then condition 2 is met. A convergence to the solution in finite time is assured. However we still need to study if forwarding along all of the paths of the solution is loop-free when using destination based hop-by-hop forwarding.

3.4 Destination based hop by hop forwarding loop free solution

In this section, we study the required conditions to assure that packets are forwarded without loops, using simple destination based hop by hop forwarding along all equally preferred paths. When the forwarding is done based on hop by hop, circuits presented in the network can cause forwarding loops (essentially because of "independent" and possibly not consistent node decisions). As described previously, nodes forward packets based on the final result of the matrix iteration, σ^k (the closure matrix of *A**) and therefore loop detection is impossible at this stage.

The problem presented in this section will use figure 3.2 as our network model. Consider node 1 as the source and node 4 as the destination. If a single path routing performing a shortest path routing algorithm is considered then no forwarding loops exist. This coherent decision can be verified by observing the closure matrix obtained by all nodes. We will have $\sigma_{1,4}^k = w(1, 5) \otimes w(5, 4)$ and $\sigma_{5,4}^k = w(5, 4)$ which provide a correct solution. However, if a shortest path routing algorithm is not in place, then a situation where the path $\sigma_{5,4}^k = w(5, 1) \otimes w(1, 2) \otimes w(2, 3) \otimes w(3, 4)$ is preferred to

the one chosen before ($\sigma_{5,4}^{k}$ = *w*(5, 4)) is perfectly acceptable. In this situation a problem may arise if the path from node 1 to node 4 maintains the preference ($\sigma_{1,4}^{k}$ = *w*(1, 5) ⊗ *w*(5, 4)), because a forwarding loop would occur between nodes 5 and 1 (packets will be trapped between those two nodes). The problem further complicates if a multipath routing protocol is being used in our network model.

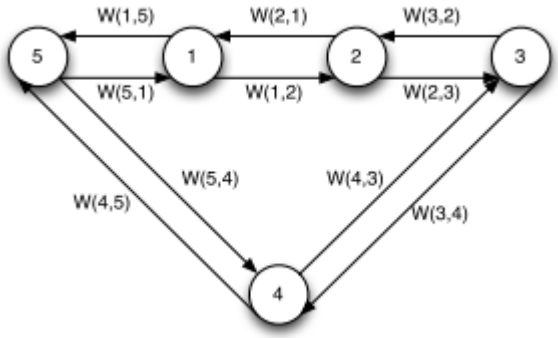


Figure 3.2: Directed graph.

The multipath case

As mentioned in chapter 2.3, in the multipath case packets can be forwarded through any of the paths of the solution minimal set. Meaning that different paths can have the same preference and therefore different forwarding decisions can occur. For example, let's assume that the minimal set of node 2 is constituted by two equally preferred paths to node 4:

$$\sigma_{2,4}^{k} = \{w(2, 1) \otimes w(1, 5) \otimes w(5, 4); w(2, 3) \otimes w(3, 4)\}$$

Also consider the minimal set of node 1 to node 4 (as well as with two equally preferred paths):

$$\sigma_{1,4}^{k} = \{w(1, 2) \otimes w(2, 3) \otimes w(3, 4); w(1, 5) \otimes w(5, 4)\}$$

In this situation a forwarding loop can happen since incoherent decisions can be made by node 2 and node 1. In the case if both nodes prefer to forward packets through the longest path solution then a loop will occur between them. This can be solved by choosing the shortest path in at least one of the nodes. However, in the multipath case all equal preference paths are part of the solution.

At this point, and using this example, a little remark must be made. Let us examine the first path, $w(C_{2,4}^{R})$ (considering $w(V_d, d)$ = *w*(4, 4)). We can observe that $w(C_{2}^{R}) \otimes w(C_{2,4}^{R})$ and $w(C_{2}^{L}) \otimes w(C_{2,4}^{R})$ do not belong to the minimal set. From condition 2 we can assure finite time convergence, since $w(C_{2,4}^{R})$ is preferred to both of them. Which means that although condition 2 assures the convergence condition, it is not enough to prevent forwarding loops. If we apply the same

reasoning to the paths in node 1 then the same result is obtained (finite time convergence guaranteed from condition 2 but still with a forwarding loop).

A typical solution to prevent forwarding loops is related to condition 2 [30], where a single path is chosen whenever a closed circuit is present. The solution is to always choose the shortest paths to leave the cycles (as mentioned in the beginning of this section).

Let us consider the generic circuit of figure 3.1, the definition 14 of directed cycles and the definition of V_d as a vertex that reaches a destination *d* by leaving the circuit. We can assure loop free forwarding if the following condition is met:

> Condition 4. Destination based hop by hop forwarding is loop free: for every circuit *C* in *G*(*V*, *E*), and all destinations *d* reached by the vertices $V_i \in C$, there is at most one vertex V_i per circuit such that:
>
> $$w(C^R_{V_i,V_d}) \otimes w(V_d,d) = w(C^L_{V_i,V_d}) \otimes w(V_d,d)$$
>
> For all other vertices V_x in *C* we have:
>
> $$w(C^R_{V_x,V_d}) \otimes w(V_d,d) \prec w(C^L_{V_x,V_d}) \otimes w(V_d,d) \forall V_x \in C^R_{V_i,V_d}$$
>
> $$\wedge$$
>
> $$w(C^L_{V_x,V_d}) \otimes w(V_d,d) \prec w(C^R_{V_x,V_d}) \otimes w(V_d,d) \forall V_x \in C^R_{V_i,V_d}$$

Which means that, in a closed circuit, it can only exist one node containing equally preferred paths to *d* via the two possible directed cycles. All the other nodes must have a preferred path to *d* according to their relative position to V_i, so that packets never return to V_i. This condition is less restrictive than the one using a single path solution and allow some of the nodes to use more than one path.

A forwarding loop only occurs if two nodes V_i and V_j belonging to *C* have a preferred path to the destination d that contains the other node. Consider that the solution in V_i is $\sigma^k(V_i,d) = w(C^R_{V_i,V_d}) \otimes w(V_d,d) \cup w(C^L_{V_i,V_d}) \otimes w(V_d,d)$ meaning that the chosen path either has the sub-path $w(C^R_{V_i,V_d})$ or $w(C^L_{V_i,V_d})$. The same happens in V_j, $\sigma^k(V_j,d) = w(C^R_{V_j,V_d}) \otimes w(V_d,d) \cup w(C^L_{V_j,V_d}) \otimes w(V_d,d)$. The loop occurs in one of two situations:

- V_i chooses the path containing $w(C^R_{V_i,V_d})$ and V_j chooses $w(C^L_{V_j,V_d})$;
- V_i chooses the path containing $w(C^L_{V_i,V_d})$ and V_j chooses $w(C^R_{V_j,V_d})$.

The first situation implies that $w(C^L_{V_j,V_d}) \otimes w(V_d,d) \prec w(C^R_{V_j,V_d}) \otimes w(V_d,d)$ with $V_j \in C^R_{V_i,V_d}$ and the second that $w(C^R_{V_j,V_d}) \otimes w(V_d,d) \prec w(C^L_{V_j,V_d}) \otimes w(V_d,d)$ with $V_j \in C^L_{V_i,V_d}$. Both situations are then in contradiction with condition 4.

Forwarding correctness

From condition 4, only a vertex V_i in *C* can have two equally preferred paths to *d* in its minimal set. All the other vertices in the directed cycle segments from V_i to V_d (in the right or left direction) can only have a preferred path, either to the left or to the right depending on their relative position to V_i . Again, consider figure 3.2 as our network model.

If we consider our V_i as being V_1 then:

$$w(C^R_{V_1,V_d}) \simeq w(C^L_{V_1,V_d})$$

and thus a forwarding loop occurs if:

$$w(C^R_{V_5,V_d}) \preccurlyeq w(C^L_{V_5,V_d})$$

or

$$w(C^L_{V_2,V_d}) \preccurlyeq w(C^R_{V_2,V_d})$$

In order to have forwarding loop free in models with a non-decreasing $\otimes$ operation the following condition must be met:

> Condition 5. In a non-decreasing $\otimes$ order semigroup all paths of the routing solution of σ lead to loop free forwarding if: for all circuits C_i in a network graph there are no two nodes V_a and V_b with $b \simeq a$ and $V_a, V_b \in C_i$ such that :
>
> $$w(C^R_{V_b,V_d}) \simeq w(C^R_{V_a,V_b}) \otimes w(C^R_{V_b,V_d})$$
>
> and
>
> $$w(C^L_{V_a,V_d}) \simeq w(C^L_{V_b,V_a}) \otimes w(C^L_{V_a,V_d})$$
>
> for all $V_d \in C_i$ for all reachable destinations *d*.

Condition 5 states that a circuit in the network graph causes a forwarding loop if at least two vertices in the circuit are connected by a path that added to the directed cycle segments between these two nodes and at least another node in the circuit maintains their preference. In an increasing $\otimes$ operation this condition is always met. The same is true in a single path forwarding

case, with a non-decreasing property since preference ties are solved by choosing the shortest path to leave the circuit.

The result of condition 5 can also be met as long as the following condition for circuit labeling is taken into account:

> Condition 6. Consider all possible circuit vertices V_d in a graph circuit *C* and two vertices V_a and V_b connected via a path with weights $w(C^R_{V_a,V_b})$ and $w(C^L_{V_b,V_a})$. Consider L_u the set values in *S* that $\otimes$ added to *u* maintain preference (as defined in condition 3). If for all *i* pairs of cycles $C^R_{V_b,V_d}$ and $C^L_{V_a,V_d}$ with equal weight u_i we have $w(C^R_{V_a,V_b}) \notin L_{u_i}$ or $w(C^L_{V_b,V_a}) \notin L_{u_i}$ then, forwarding is loop free.

From this condition we can conclude that an especially care must be taken in the choice of the policy/weight. Otherwise, a lot of possible circuit labelings can result in forwarding loops. Although the condition 6 assures loop free destination based hop by hop forwarding, it imposes a severe restriction to $L_u$sets and it is hard to verify. We therefore find a condition that assures condition 5 for all possible circuit labelings.

> **Definition 16**. Consider an edge/link *l* between two vertices/nodes of a network graph, *a* and *b*. Also consider the following notation: [*X*-*Y*] with *X* and *Y* belonging to the set *S* meaning that *l* has a weight *X* from *a* to *b*, *w*(*a*, *b*) = *X*, and in the opposite direction, from *b* to *a*, the weight *w*(*b*, *a*) = *Y*. A label pair is a valid combination [*X*-*Y*] in a given set *S*: In the particular case where *X* = *Y* we say that a symmetric policy occurs in *S* (i.e. the edge has the same policy in both directions.

Having the notion of label pair in mind as well as condition 6, a new condition can be derived:

> Condition 7. Consider all *X*, *Y* labels in *S*. A forwarding loop can occur whenever a [*X*-*Y*] label pair belongs to one of the L_u sets (condition 3):
>
> $\forall$ *X*, *Y* $\in$ *S* forming a label pair [*X*-*Y*], *X*, *Y* $\nexists L_u \forall L_u$

Proof: Consider the case illustrated in figure 3.3 where a label pair [*a*-*b*] is applied to the directed edges connecting nodes 0 and 1. Also consider that nodes 1 and 2 are connected by a path with weight *u* (which can be a single link path or a multiple link path). From condition 3 we know that *a* and *b* only belong to set L_u if their $\otimes$ addition to the path weight *u* maintains preference, which means: $a \otimes u = b \otimes u = u$. If this is the case (i.e. $\{a, b\} \in L_u$) then, the path between nodes 0 and 2 (0, 1, 2) will have weight *u* as well as the path (1, 0, 2) and therefore a forwarding loop can occur among node 0 and node 1. But if $a \notin L_u$ and $\otimes$ is non-decreasing then $w(u) \prec w(a) \otimes w(u)$ and condition 5 is met. The same reasoning can be made for if $b \notin L_u$ meaning that if either one is true, forwarding loops do not occur.

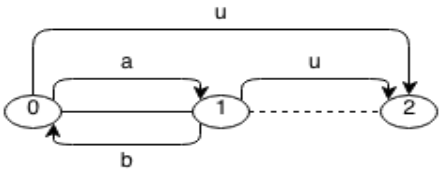


Figure 3.3: Forwarding loop problem illustration.

This results means that if Condition 7 holds in an order semigroup with a non-decreasing ⊗ operation then forwarding packets, in destination based hop by hop, is assured without causing forwarding loops without any restrictions in circuit labelings.

3.5 An example generic multipath policy routing protocol

In this section a generic multipath routing protocol based in policies from [1] is in part described and is used to exemplify the application of the conditions stated so far in this thesis and that will be automatized with the developed software.

3.5.1 Protocol algebraic model

The concept of hierarchy fits reasonably well in the way networks are organized. There is a first separation between access/border areas formed by routers providing connectivity to hosts or to routers in other networks, and transport/backbone areas formed by routers that interconnect the border/access routers to transport traffic. Within the transport/backbone area some hierarchical levels can also be present. This hierarchy can be based on node degree (i.e. number of links connecting a node), geographic location, economic relationships or any other differentiating characteristic.

Therefore, given the concept of hierarchy, the example model defines a set of policies based on the relative position between two nodes in the network hierarchy. Paths are qualified with a weight depending on the direction of the hierarchy they follow. By using policies, several paths can end up having equivalent weights and behave similarly in the hierarchy, but can be different in all other aspects (i.e. number of hops, links, capacity, or bandwidth).

The network is modeled as described in section 3.2 and policies are expressed by values $l \in S$ and each edge $e \in E$ is labeled by a value l. Note that the way values are assigned to links creates a specific path distribution for each network graph with the consequences of having more or less path diversity. Also, the definition of the set of values in S and the way they are composed with ⊗ defines the possible types of paths and how traffic can be distributed in the network.

Hierarchies are two-dimensional structures and so three elements in *S* are enough to express all the policies:

D_W – links in the downwards direction of the hierarchy.

U_W – links in the upwards direction.

S_L – links between nodes at the same level of hierarchy.

A link between two nodes at different levels of the hierarchy is modeled by two directed edges in the graph: one in the upward direction (U_W) and another in the downward direction (D_W). If the link connects nodes at the same level of the hierarchy (peer node) both directed edges are labeled with the same value (S_L). An analogy between these policies and the well-known policies used in the inter-domain scenario (mentioned in section 2.3.5) can be made: the D_W policy is comparable to the *p2c* (provider-to-costumer) policy; the S_L can be seen as the *p2p* (peer-to-peer) policy; and U_W is similar to *c2p* (costumer-to-provider) policy.

Trivial paths are modeled by $\overline{1}$ corresponding to the highest preference possible and invalid paths which have the lowest preference are modelled by $\overline{0}$. So we have

$$S = \{\overline{1}, D_W, S_L, U_W, \overline{0}\} \qquad \text{(Eq. 3.3)}$$

and the path preference is modeled by the preference order ,

$$\overline{1} \preccurlyeq D_W \preccurlyeq S_L \preccurlyeq U_W \preccurlyeq \overline{0} \qquad \text{(Eq. 3.4)}$$

meaning that D_W paths have the highest preference followed by peer S_L paths and finally U_W paths have the lowest preference. So, going downwards in the hierarchy is more preferred than going upwards. Another important feature of a protocol, and which will help to verify the conditions stated, are the label pairs (definition 16). In this specific protocol such label pairs are:

$$[\overline{1}\text{-}\overline{1}],\ [D_W\text{-}U_W],\ [S_L\text{-}S_L] \text{ and } [\overline{0}\text{-}\overline{0}]. \qquad \text{(Eq. 3.5)}$$

Meaning that if a link between two nodes (for example a node *a* and a node *b*) has a weight of D_W from *a* to *b*, then the weight from *b* to *a* has to be U_W. Note however, that $[D_W\text{-}U_W]$ implies $[U_W\text{-}D_W]$.

The elements of *S* and the order relation can be interpreted in abstract terms without any meaning to what a “lower hierarchical level” is in a concrete network or how a node should be considered of higher level than another. However, they define how paths are calculated and how traffic can flow in the network, and the hierarchy relationships between the nodes should be chosen accordingly to the desired behavior.

3.5.2 Protocol implementation model

This subsection introduces the model of the algorithm used to calculate the routing solution. The algorithm calculates the routing solution and is used through the rest of this dissertation. Its comprehension will be important to better understand section 3.5.3. The first step in the algorithm is to obtain a representation of the network topology.

The adjacency matrix *A* (eq. 2.3) is the algebraic representation of a network graph/topology and is always a square matrix. Consider *P*(*i*, *j*) the set of all paths between a source *i* and a destination *j*. Their weights are given by *w*(*P*) and belong to the set *S* as shown in equation 3.3.

The identity matrix *I* has the same size of matrix *A*, and the main diagonal only contains the trivial path with value $\overline{1}$ (representing the most preferred path) and the remaining signatures are labelled with unreachable value $\overline{0}$ (representing the less preferred),

$$I = \begin{bmatrix} \overline{1} & \overline{0} & \dots & \overline{0} \\ \overline{0} & \overline{1} & \dots & \overline{0} \\ \vdots & \vdots & \ddots & \vdots \\ \overline{0} & \overline{0} & \dots & \overline{1} \end{bmatrix}$$

The routing solution can be found by solving the matrix equation 3.1, also called left routing problem equation, and corresponds to the closure matrix of adjacency matrix *A**. As said, the left routing solution calculates the path from destination to source, which prevents loops in a local routing solution (since monotonicity is not present). Matrix *A** may be obtained via matrix iteration [35], corresponding to the following matrix equation (which is related to eq. 3.2):

$$A^{k+1} = A^* \otimes A^k \text{ for } K \in \mathbb{N} \qquad \text{(Eq. 3.6)}$$

The identity matrix *I* is only used in the first iteration, meaning we start with the identity matrix *I* and left $\otimes$ multiply *A* recursively. *A** is obtained if equation 3.6, reaches a convergence solution in a finite number of iterations, i.e. when it verifies the condition,

$$A^k = A^{k+1} \text{ for } K \in \mathbb{N} \qquad \text{(Eq. 3.7)}$$

The following example will help to understand how the matrix multiplication works. Consider the graph in figure 3.4 with two levels of hierarchy.

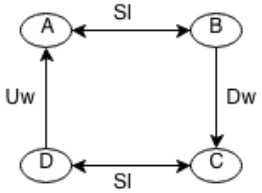


Figure 3.4: Simple hierarchy graph.

In the example of figure 3.4, nodes *D* and *C* rise in the hierarchy (U_W) to reach nodes *A* and *B*, respectively. On the other hand, and since [D_W-U_W] is a label pair of set *S*, nodes *A* and *B* descend in the hierarchy (D_W). Nodes *A* and *B* are at the same level (S_L), as well as nodes *C* and *D*. The adjacency matrix is:

$$A = \begin{bmatrix} \overline{1} & S_L & \overline{0} & D_W \\ S_L & \overline{1} & D_W & \overline{0} \\ \overline{0} & U_W & \overline{1} & S_L \\ U_W & \overline{0} & S_L & \overline{1} \end{bmatrix}$$

Now, to find the routing solution, eq. 3.6 is used. Take for instance the calculation for the position (1, 2) of a generic matrix A with n elements.

$$a_{12}^{k+1} = (a_{11} \otimes a_{12}^{k}) \oplus (a_{12} \otimes a_{22}^{k}) \oplus \ldots \oplus (a_{1n} \otimes a_{n2}^{k})$$

This means that to obtain the most preferred weight for a path with *k* edges from origin 1 to destination 2 (a_{12}^{k+1}) it is necessary to compose all possible paths to the destination using the $\otimes$ operation path composition and then choosing the most preferred values using the $\oplus$ operation path selection.

The path composition operation calculates various paths weights extending one path at a time. For the aforementioned example the first path is composed by the origin (1) and a next hop (n); and the second is the path from the next hop (n) to the destination (2). Since the eq. 3.6 solves the multipath problem, in the path selection operation all equally preferred next hops are selected.

Let's take the example from the topology of figure 3.4, using matrix *A** to calculate the value of the path from node *A* to nodes *B* and *C*. The path composition operation $\otimes$ is the one defined in the next section, table 3.2.

$$A_{(A,C)}^{2} = (A_{(A,A)} \otimes A_{(A,B)}) \oplus (A_{(A,B)} \otimes A_{(B,B)}) \oplus (A_{(A,C)} \otimes A_{(C,B)}) \oplus (A_{(A,D)} \otimes A_{(D,B)})$$

$$= (\overline{1} \otimes S_L) \oplus (S_L \otimes \overline{1}) \oplus (\overline{0} \otimes U_W) \oplus (D_W \otimes \overline{0})$$

$$= S_L \oplus S_L \oplus \overline{0} \oplus \overline{0}$$

$$A_{(A,C)}^{2} = (A_{(A,A)} \otimes A_{(A,C)}) \oplus (A_{(A,B)} \otimes A_{(B,C)}) \oplus (A_{(A,C)} \otimes A_{(C,C)}) \oplus (A_{(A,D)} \otimes A_{(D,C)})$$

$$= (\overline{1} \otimes \overline{0}) \oplus (S_L \otimes D_W) \oplus (\overline{0} \otimes \overline{1}) \oplus (D_W \otimes S_L)$$

$$= \overline{0} \oplus S_L \oplus \overline{0} \oplus \overline{0}$$

Sorting by preference using the path selection operation, eq. 3.4, we are left with:

$$A^2_{(A,B)} = S_L \text{ and } A^2_{(A,C)} = S_L$$

The process of eq. 3.6 is repeated to solve the entire matrix until it reaches a stop condition, eq. 3.7. The solution for the example of figure 3.4 is:

$$A^2 = \begin{bmatrix} \overline{1} & S_L & S_L & D_W \\ S_L & \overline{1} & D_W & S_L \\ U_W & U_W & \overline{1} & S_L \\ U_W & U_W & S_L & \overline{1} \end{bmatrix}$$

If we iterate one more time than the result is $A^2 = A^3$, and therefore the topology solution is found (eq. 3.7). The next hop table for the topology is:

$$\begin{bmatrix} - & B & B & D \\ A & - & C & A \\ B & B & - & D \\ A & A & C & - \end{bmatrix}$$

The next hop table is filled during the path selection operation. Note, that from node *A* to node *B* the preferred paths are from nodes (*A*, *B*), but since node *A* cannot be the next hop (to itself), the next hop for destination *B* is in fact only {*B*} (node *B*). An important observation to make is that that using table 3.2 as the ⊗ operation in this specific network then only one path is used from all nodes to all destinations. In this situation there is no multipath.

The inexistence of multiple paths is not due to the algorithm itself but because of the specific topology and the definition of ⊗. Let us see another brief example that confirms this statement. Instead of using table 3.2 as our ⊗ operation we will now use table 3.1 (also present in the next section), and we will continue to consider figure 3.4 as our network graph model. In this scenario our adjacency matrix *A* is exactly the same as before:

$$A = \begin{bmatrix} \overline{1} & S_L & \overline{0} & D_W \\ S_L & \overline{1} & D_W & \overline{0} \\ \overline{0} & U_W & \overline{1} & S_L \\ U_W & \overline{0} & S_L & \overline{1} \end{bmatrix}$$

Using source node (*A*) and destination node (*C*), we now have:

$$A^2_{(A,C)} = (A_{(A,A)} \otimes A_{(A,C)}) \oplus (A_{(A,B)} \otimes A_{(B,C)}) \oplus (A_{(A,C)} \otimes A_{(C,C)}) \oplus (A_{(A,D)} \otimes A_{(D,C)})$$

$$= (\overline{1} \otimes \overline{0}) \oplus (S_L \otimes D_W) \oplus (\overline{0} \otimes \overline{1}) \oplus (D_W \otimes S_L)$$

$$= \overline{0} \oplus S_L \oplus \overline{0} \oplus S_L$$

Following the same steps, described in the previously example, the solution (the closure matrix) will be:

$$A = \begin{bmatrix} \overline{1} & S_L & S_L & D_W \\ S_L & \overline{1} & D_W & S_L \\ U_W & U_W & \overline{1} & S_L \\ U_W & U_W & S_L & \overline{1} \end{bmatrix}$$

And the next hop table is:

$$\begin{bmatrix} - & B & \{B,D\} & D \\ A & - & C & \{A,C\} \\ \{B,D\} & \{B,D\} & - & D \\ \{A,C\} & \{A,C\} & C & - \end{bmatrix}$$

So, in this table, the next hops for destination node *C* from source node *A* are {*B*, *D*}. In this case we have a multipath solution. Note, that this solution is only possible due to the use of an algorithm that stops the iteration when travelling around cycles, thus allowing convergence in finite time. However, and as we will see next, this solution contains forwarding loops and therefore the correct operation of the protocol is not assured (at least in this network graph).

The correct behavior will dependent on the ⊗ operation the *S* set and the order algebraic properties and how the labels in *S* are applied in the network. This is the main focus of this thesis and will be the theme of the next section, where the tradeoffs between the possible network labelings, the flexibility of the applicable preferences and the number of usable paths is discussed.

3.5.3 The ⊗ operation flexibility and its impact on the model

The ⊗ operation defines how paths are calculated defining how traffic can be distributed in the network. Since we can have a non-decreasing ⊗ operation, to increase the number of paths in the routing solution, some restrictions have to be made from start since such an operation does not assure by itself a correct protocol operation. So, in order to guarantee finite time convergence and loop free forwarding (definition 1) some conditions presented in 3.3 and 3.4 must be met. These conditions, as we will see, restrict the possible results of the ⊗ operation.

In this section, the algebraic protocol model presented in 3.5.1 is used and different ⊗ operations are studied. The intention is to illustrate the balance between the ⊗ operator flexibility and the restrictions imposed by the correctness conditions. We start by the most general ⊗ operation possible in our example model, where invalid paths are not presented, and then we evolve to the correct and final solution presented in [1].

The ⊗ operation without invalid paths

⊗	$\overline{1}$	D_W	S_L	U_W
$\overline{1}$	$\overline{1}$	D_W	S_L	U_W
D_W	D_W	D_W	S_L	U_W
S_L	S_L	S_L	S_L	U_W
U_W	U_W	U_W	U_W	U_W

Table 3.1: ⊗ operation without invalid paths.

First, using condition 3 to analyze the possible non-free cycles, we start by defining the sets L_u for each $u \in S\backslash\overline{0}$. From table 3.1 we have the following sets:

- $L_{\overline{1}} = \{\}$ because there is no element $l \in S$ such that $\overline{1} = l \otimes \overline{1}$
- $L_{D_W} = \{D_W\}$ since only for $l = D_W$ we have $D_W = l \otimes D_W$
- $L_{S_L} = \{D_W, S_L\}$ because $S_L = D_W \otimes S_L$ and $S_L = S_L \otimes S_L$
- $L_{U_W} = \{D_W, S_L, U_W\}$ since $U_W = D_W \otimes U_W$, $U_W = S_L \otimes U_W$ and $U_W = U_W \otimes U_W$

Remember, from section 3.3 that if an algorithm does not stop when travelling around cycles, then finite time convergence is not assured if all links in a cycle are labeled with values belonging to one of the sets L_S (at least one). In this case L_{U_W} contains all possible link values, which causes all cycles to be non-free (Condition 2).This means that the network would have to be acyclic and therefore without multipath.

Take the example, illustrated in figure 3.5, of two disjoint[10] paths leaving from a source at a high level in the hierarchy (*A*) towards a node at a lower level (*C*), and from there to the destination (*d*). All of the disjoint paths could not exist simultaneously because they imply a cycle in the network in which all links belong to L_{U_W}.

[10]A disjoint path is commonly referred to paths (between two nodes) where no link is common. Whereas, for different paths the presence of only one different link is enough.

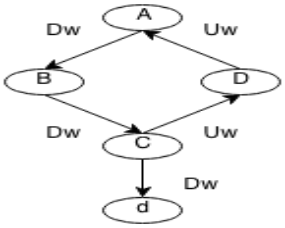


Figure 3.5: Simple network example.

A way to solve this problem is by changing the algebra, such that some of the L_S sets have fewer elements (as showed in next section). The solution is to invalidate some paths in the $\otimes$ operation and thus eliminating the problematic sets.

Note, however, that if the algorithm stops the iteration when travelling around circuits then finite time convergence to a stable routing solution is assured (as stated in section 3.3). Although this solution solves the problem mentioned above, it does not guarantee the correct operation of the protocol since it does not assure loop free forwarding. Consider for example the label pair [D_W-U_W], that is applied in the network of figure 3.5 (e.g. between node *A* and node *B*). From condition 7 we know that if this label pair belongs to any of the sets L_S a forwarding loop can occur. Since, in this example, the problematic set L_{U_W} contains both the weights presented in the label pair, forwarding loop freeness is not assured.

To illustrate this forwarding problem, take as an example the routing solution from node *B* to node *D*. Node *B* will have in its routing table two possible paths to reach node *D*, the one through *C* ($D_W \otimes U_W$) and the one through *A* ($U_W \otimes D_W$), since both result in a path with U_W weight. Node *C*, in its turn, also contains two paths to *D*, the one directly to *D* (U_W) and the one through *B* and *A* ($U_W \otimes (U_W \otimes D_W) = U_W$). In a multipath perspective this is not bad at all however, a problem arises if B chooses to forward packets to *D* through *C* and *C* chooses the longest path (using the path through *B* and *A*) to forward packets to *D*. If this happens, packets can be trapped between nodes *B* and *C* performing a forwarding loop.

In this example forwarding correctness is not assured because the set L_{U_W}contains both the D_W and the U_W weights of the label pair [D_W-U_W].

The ⊗ operation with invalid entries

⊗	$\overline{1}$	D_W	S_L	U_W
$\overline{1}$	$\overline{1}$	D_W	S_L	U_W
D_W	D_W	D_W	$\overline{0}$	$\overline{0}$
S_L	S_L	S_L	S_L	$\overline{0}$
U_W	U_W	U_W	U_W	U_W

Table 3.2: ⊗ operation with invalid paths.

The previously section demonstrate that if a set L_S contains all links then finite time convergence and loop free forwarding are not possible. So, the solution can pass by invalidating some operations so that the sets L_S are reduced:

- $L_{\overline{1}}$ and L_{D_W} remains equal
- L_{S_L} = $\{S_L\}$ given, that now $\overline{0}$ = $D_W \otimes S_L$
- L_{U_W} = $\{U_W\}$ given, that now $\overline{0}$ = $D_W \otimes U_W$, $\overline{0}$= $S_L \otimes U_W$

The invalid combinations of links resulting in weight $\overline{0}$ exist to reduce the number of possibilities where conditions 3 and 7 are broken. With the above sets condition 3 is met and therefore finite time convergence to a stable routing solution is assured.

But, under the light of condition 7, a problematic set is still in present, the set L_{S_L}, because it contains the label pair [S_L-S_L].

Figure 3.6: Network example.

To illustrate the problem, take as an example the network presented in figure 3.6. Let's consider the case from node *B* to node *C*. In the first matrix iteration node *B* only knows that to reach *C* it has to send packets through the direct link. However, in the third iteration, the *B-A* link will be ⊗ added to path *A-B-C* (that results from the second iteration) resulting in a path *B-A-B-C* and

therefore causes a forwarding loop. Since forwarding is based on destination hop-by-hop, node *B* will never imagine that for *A* to reach *C* it will be thrown to himself. Node *B* will therefore compute its routing table solution which will contain, in its next hop table, two possible paths to node *C* (through nodes *A* and *C*). In this way a forwarding loop can occur between nodes *A* and *B* when node *C* is the destination. This is mainly due to the result of $S_L = S_L \otimes S_L$ which leads to the presence of S_L in the set L_{S_L}, which is strictly related to condition 7.

In resume, for this $\otimes$ operation, correctness is still not assured since definition 1 is not met. This conclusion is in agreement with the one provided by the author of [1], where some extra conditions are applied to the protocol in order to assured its correctness. The solution adopted was to change the algebra so that adding a S_L link has consequences in terms of preference depending on the direction travelled. A sense of direction in S_L links was created. With this paths traversing nodes at the same level have different preferences according to the different directions. This solution solves both the path calculation problem and the forwarding loops since paths do not have any more the same preference in both directions of the cycle. In practical terms a new attribute was added to the S_L links that denote a secondary preference. Thus, it is modelled by a secondary algebra, since generically, two different routing metrics can be modelled in an algebraic form by taking the lexicographic product of the two algebras (one for each metric) [35] [49].

3.6 Generalizing the design of a multipath routing protocol

Through this chapter we studied the correct properties that a multipath routing solution must have in order to assure correct operation and an example protocol to illustrate the concept is presented. We now use these concepts to design a routing protocol with a generic policy set *S*. The idea is to abstract the needed steps and illustrate the design variables involved.

The elements of the protocol set *S*, usually representing a link/path weight, are now expressed in the most abstract way possible, meaning they can cover the policy part (definition 2) of a protocol from a simple numeric metric to a complex policy attribute. Note, that this abstraction can only be made because algebraic structures are in use, showing their great potential to model such protocols.

The correct behavior of a multipath policy based protocol can be assure by condition 3 and condition 7. Condition 3 assures finite time convergence whereas condition 7 guarantees forwarding loop correctness using destination based hop-by-hop forwarding.

A more mindful reader can perhaps conclude that in order to meet condition 7, condition 3 has always to be met, which is correct. This conclusion becomes even truer when an algorithm that stops iteration in the presence of circuits (definition 13) is used (as we do). However, for the sake of having a more granular verification of the protocol, both conditions will continue to be used. This is important since it reveals in an easier way the drawbacks of a certain protocol, allowing to make a distinction between finite time convergence problems and forwarding loop problems.

3.6.1 Protocol set model

The protocol model can be represented by a set *S* with *n* elements, provided that $n \in \mathbb{N}$. However, and in order to have a simplistic model (such as: a finite number of set *S* elements; and an easier representation of label pairs), our generic set *S* will be represented by:

$$S = \{a, b, c, d, e, f, g\} \qquad \text{(Eq. 3.8)}$$

where the preference order is:

$$a \preccurlyeq b \preccurlyeq c \preccurlyeq d \preccurlyeq e \preccurlyeq f \preccurlyeq g \qquad \text{(Eq. 3.9)}$$

and the label pairs (definition 16) are:

$$[a\text{-}a] \wedge [b\text{-}f] \wedge [c\text{-}c] \wedge [d\text{-}e] \wedge [g\text{-}g] \qquad \text{(Eq. 3.10)}$$

An important aspect to retain is that the model should always contain the elements representing the weights of the non-existent or invalid path, in this case expressed as *g*, and the weight of the trivial path, symbolized by *a* (respectively associated to the $\overline{0}$ and $\overline{1}$ used before).

The label pairs were assigned having in mind the following considerations: the [*a*-*a*] and [*g*-*g*] are intuitive, since for both trivial paths and invalid paths the correspondent symmetrical link must also have the same weight (therefore, they can be considered a characteristic of our protocol model); the [*b*-*f*] was defined having in mind the objective of having a relation between the best and worst paths (apart from the trivial and invalid paths), which may be used as links going up or down in a hierarchical topology (an analogy can be made with the label pair [D_W-U_W] presented in section 3.5); the [*c*-*c*] was defined in order to have possible symmetrical links independently of their direction, which may have its utility between nodes at the same hierarchy level (i.e. the label pair [S_l-S_l] used in the previous section); at last the [*d*-*e*] was defined aiming to give a possible sense of direction (therefore eliminating the necessity of a second algebra, like the one used in the solution mentioned in the end of section 3.5).

The way values are assigned to links creates a specific path distribution for each network graph with consequences to having more or less multiple paths (path diversity). Also, the definition of the set of values in *S* and the way they are composed with the ⊗ operation defines the possible types of paths and how traffic can be distributed in the network.

3.6.2 ⊗ Definition

The ⊗ operation defines how paths are calculated and how traffic can flow in the network. Let us then define a non-decreasing ⊗ operation based on the conditions for correct behavior (do not forget that non-decreasing is the least restrictive property for multipath).

In order for ⊗ to be non-decreasing (definition 8) and according to the order definition the following table contains all possible results of ⊗. Table 3.3 represents the addition of a link *l* (illustrated in vertical in the left column) to path/link *u* (illustrated in horizontal in the first row). The other labels represent the resulting path of such combination, for example *b* ⊗ *c* can result in one of these elements {*b*, *c*, *d*, *e*, *f*, *g*}:

⊗	*a*	*b*	*c*	*d*	*e*	*f*
a	*a,b,c,d,e,f,g*	*b,c,d,e,f,g*	*c,d,e,f,g*	*d,e,f,g*	*e,f,g*	*f,g*
b	*a,b,c,d,e,f,g*	*b,c,d,e,f,g*	*c,d,e,f,g*	*d,e,f,g*	*e,f,g*	*f,g*
c	*a,b,c,d,e,f,g*	*b,c,d,e,f,g*	*c,d,e,f,g*	*d,e,f,g*	*e,f,g*	*f,g*
d	*a,b,c,d,e,f,g*	*b,c,d,e,f,g*	*c,d,e,f,g*	*d,e,f,g*	*e,f,g*	*f,g*
e	*a,b,c,d,e,f,g*	*b,c,d,e,f,g*	*c,d,e,f,g*	*d,e,f,g*	*e,f,g*	*f,g*
f	*a,b,c,d,e,f,g*	*b,c,d,e,f,g*	*c,d,e,f,g*	*d,e,f,g*	*e,f,g*	*f,g*

Table 3.3: Protocol model ⊗ operation 1.

From table 3.3, it is easy to see that a non-decreasing ⊗operation presents a pattern where the possible results are reduced when ⊗ adding links to less preferred labels.

Another important consideration that impacts the verification of conditions 3 and 7 and therefore the correctness of the protocol is the size of the L_u sets. This means that special attention must be taken in the choice between the possible ⊗ results.

In the following table we reduce the possible results so that conditions 3 and condition 7 are taken into account:

⊗	*a*	*b*	*c*	*d*	*e*	*f*
a	*a*	*b*	*c*	*d*	*e*	*f*
b	*b*	*b,c,d,e,f,g*	*c,d,e,f,g*	*d,e,f,g*	*e,f,g*	*f,g*
c	*c*	*c,d,e,f,g*	*d,e,f,g*	*e,f,g*	*f,g*	*g*
d	*d*	*b,c,d,e,f,g*	*c,d,e,f,g*	*d,e,f,g*	*e,f,g*	*f,g*
e	*e*	*b,c,d,e,f,g*	*c,d,e,f,g*	*d,e,f,g*	*e,f,g*	*f,g*
f	*f*	*b,c,d,e,f,g*	*c,d,e,f,g*	*d,e,f,g*	*e,f,g*	*f,g*

Table 3.4: Protocol model ⊗ operation 2.

Since the abstract element *a* represents the trivial path (which should always be present) in the first column the results of adding any label to *a* results is that label: *l* = *l* ⊗ *a*, and the first row the values reflect the fact that any label to which *a* is added results in that same label. This is because the trivial path label is a neutral element of ⊗.

We now take in consideration condition 7 that forbids a set L_u to contain both values of a label pair. That said, from the label pairs definition it is easy to see that element *c* cannot be present in any of the sets L_u, since it forms a label pair with himself [*c*-*c*] (meaning that a link with "weight" *c* has the same value in both direction). So, and to prevent this problem, the row of element *c* takes in to account that for all *u* in *S* $u \neq c \otimes u$ thus excluding element *c* from every set L_u.

We can see in table 3.4 that keeping the ⊗ operation within the limits of condition 7 reduces the possible results.

For example: if we decide that adding a link with weight *b* to a path with the same weight *b* results in a path with a weight *b* ($b \otimes b = b$) then *b* is an element of set L_b. Since the label pair of *b* is *f*, [*a*-*f*], we cannot have element *f* in set L_b, meaning that adding a link with weight *f* to a path with weight *b* cannot result in a path with weight *b*. To assure this $f \otimes b \neq b$ has to be true.

In summary, several different combinations of the ⊗ operation are possible and still the correct operation of the protocol is maintained. In table 3.5 a possible non-decreasing ⊗ operation solution is presented that verifies conditions 3 and 7:

⊗	*a*	*b*	*c*	*d*	*e*	*f*
a	*a*	*b*	*c*	*d*	*e*	*f*
b	*b*	*b*	*g*	*g*	*g*	*g*
c	*c*	*c*	*g*	*g*	*g*	*g*
d	*d*	*d*	*d*	*d*	*g*	*g*
e	*e*	*e*	*e*	*e*	*e*	*g*
f	*f*	*f*	*f*	*f*	*f*	*f*

Table 3.5: Protocol model ⊗ operation 3.

To prove this we start by defining the sets L_u for each $u \in S\backslash\overline{0}$:

- L_a = {} because there is no element $l \in S$ such that $a = l \otimes a$
- L_b = {b} since only for $l = b$ we have $b = l \otimes b$
- L_c = {} since there is no element $l \in S$ such that $c = l \otimes c$
- L_d = {d} since only for $l = d$ we have $d = l \otimes d$
- L_e = {e} because only for $l = e$ we have $e = l \otimes e$
- L_f = {f} since only for $l = f$ we have $f = l \otimes f$

Remember from condition 3 that finite time convergence is compromised if there is at least a set L_u containing all the links of set S and from condition 7 that forwarding loops can occur if there is a set L_u containing both the elements of a label pair.

As it is easy to see, from the L_u sets that both conditions are obeyed and therefore protocol correctness is assured.

3.6.3 Summary

From the generalization above some conclusions can be drawn. A first aspect is that there is a trade-off between the amount of possible equally preferred paths (that depends on the path composition operation and preference order) and the network restrictions needed for the protocol to work correctly. To increase the possibility to have equally preferred paths the $\otimes$ operation has to be merely non-decreasing, and this implies that the network has to be restricted to not having non-free cycles.

In order to assure correctness without posing restrictions on network topology, some paths have to be considered invalid and the way to calculate paths must have the need for small sets of preference maintaining labels in to account. This means that we can only use the network path diversity to some extent. Another aspect is the way traffic behaves given the characteristics of the algebraic model and the defined preference order. It is important to understand these characteristics to perform the labeling of the links in the network in order to achieve the best possible use of the network's path diversity.

The next chapter describes the software tools developed through this dissertation to verify these design constraints automatically. We also describe the implementation in software of the algorithm presented above to calculate the routing solution for a given protocol. These software tools receive as an input a given network graph which is described by an adjacency matrix that can be automatically generated, or inserted by the user. It receives a description of the algebraic model of a routing protocol and presents the user with results that show if it will operate correctly or not and what conditions or properties are verified or not. Finally they allow the calculation of the routing solution for several different combinations of edge labelings. We use the software to perform several experiments to study the characteristics outlined in this chapter and provide examples of the software's utility in the policy based multipath routing protocols study and design.

CHAPTER 4: Performance Analysis

4.1 Introduction

This chapter describes the software developed during the dissertation, followed by some experiments that study specific protocols and also serve as an example of the utility of the developed tools for protocol verification and design.

The software was primarily designed to analyze and verify the correct properties and conditions that a multipath routing protocol, modeled by an algebraic structure, must have in order to assure its correctness (accordingly to the conditions presented in chapter 3). However, the necessity to test and confirm the conditions stated lead to some extensions. These extensions provide the possibility to test multipath protocols models with different topologies, and calculate the routing solutions. We can then analyze the routing solution, which contains the paths between any pair of nodes (source-destination). The presence of finite time convergence or forwarding loops problems are then easily identified. In summary, the software provides a way to confirm the conditions defined in the previous chapter.

The chapter is divided in two distinct sections. The first section introduces the software, along with its functionalities, and some implementation details and the second presents some experiments.

Section 4.2 describes the implementation of the software describing all the principal features. In 4.2.1 the used network graph generator is explained. The section 4.2.2 explains how the protocol model could be implemented and in 4.2.3 we define how protocol policies can be assigned to links. The last section of the software subchapter describes the matrix iteration algorithm and how the path diversity of the routing solution is measured.

Section 4.3 presents the topologies used in the experiments as well as some results. Section 4.3.1 presents the results for a first set of topologies, which are based on highly hierarchical and regular networks graphs. Finally, in section 4.3.2, an experiment using a scale-free network (which in general presents the same characteristics observed in an Internet inter AS network) is performed.

4.2 Software

The software presented in this section can be seen as a tool to verify, and aid in the design of multipath routing protocols modeled by algebraic structures.

In this subchapter the fundamental parts constituting the software are explained.

4.2.1 Network graph generator

In this section, a brief introduction to a software package based on the Python language, called NetworkX, is made. Its utility is of extremely importance since it allows us to create the topologies used in experiments. NetworkX is also capable to analyze some important characteristics of network graphs, a feature that is also explored in our software (as we will see ahead).

The network topologies are introduced into our software by a file (a .txt file) containing their adjacency matrix. However, using NetworkX, network graphs (our topologies) may be generated in a more simple and rapid way. This feature also allows to create an enormous variety of network graphs, from regular to scale-free topologies (which are used in the experiments section), providing the confirmation of conditions made in chapter 3 (by experimenting different protocols with different network graphs), which is a great advantage.

In this work, some topologies characteristics, obtained with NetworkX, are used, such as:

- Degree centrality: The total number of connections of a node. Thus, the larger the degree, the "more important" the node is in a network;
- Betweenness centrality: The number of shortest paths from all vertices to all others that pass through a node. A node with high betweenness centrality has large influence in the transfer of items through the network, under the assumption that transfer follows shortest paths;
- Eigenvector centrality: A measure of the influence of a node in a network. It assigns relative scores to all nodes in the network based on the concept that connections to high-scoring nodes contribute more to the score of the node in question than equal connections to low-scoring nodes (Google's PageRank is a variant of the eigenvector centrality measure).

In graph theory and network analysis, centrality refers to indicators that identify the most important vertices within a graph. Our software uses those values for automatic link weight attribution, alongside with a defined protocol set. In section 4.2.3 a better explanation of this feature is explained.

4.2.2 Protocol specification and verification

This is the main feature of the software. In order to define a certain multipath protocol model and to verify its correct operation a program written in C++ was developed.

A protocol model can be defined in a .txt file, which has to contain the following information: the definition of the protocol set *S* (what are the elements that form the set); the set order preference (which elements are preferred and which are less preferred); the label pairs of the set; and the $\otimes$

operation. Or, in other words, the fields that must be present in the text file are respectively related to: eq. 3.8; eq. 3.9; eq. 3.10; and table 3.5.

Once the protocol model is established the program can verify its properties and conditions. The properties verification serve to understand the nature of the protocol, analyzing whether a protocol is monotone (strictly or simple monotone) or not, and if the $\otimes$ operation is increasing, non-decreasing, or decreasing. The verification follows the properties equations provided in section 2.3.4. If the program concludes that the protocol is modeled by a non-decreasing $\otimes$ operation than the calculation of the L_u sets is performed and the conditions obtained in sections 3.3 and 3.4 are verified. The program states if condition 3 and condition 7 are met.

In order to understand the correct operation of the program, let us use the models presented in chapter 3. First we will model the protocol described in section 3.6.1 with the $\otimes$ operation in table 3.5. Table 4.1 shows the program results:

Protocol properties				Protocol conditions	
Strictly Monotone	Simple Monotone	Strictly Increasing	Non-Decreasing	Finite Time Convergence	Forwarding Loops
$x \prec y \Rightarrow k \otimes x \prec k \otimes y$		$y \prec x \otimes y$		Condition 3	Condition 7
The properties/conditions fails in:					

x	y	k	x	y	Set L_u	Label Pair
a	*d*	*e*	*a*	*b*		
a	*b*	*d*	*d*	*d*		
b	*c*	*d*	*e*	*e*		
c	*d*	*d*	*f*	*f*		
d	*e*	*c*	*a*	*c*		
...	...	...	...	...		

Table 4.1: Protocol program verification, from table 3.5 (Note: For the non-strict expressions $\prec$ must be changed by $\preccurlyeq$).

Table 4.1 shows some of the results provided by the program. A cross is present whenever the conditions hold and therefore means that the protocol assures the property/condition in question. In this case the program reveals that the protocol has a simple monotonic property as well as a non-decreasing property. Since the non-decreasing property is validated then the conditions verification are performed, which concludes that the protocol assures finite time convergence and forwarding loop freeness. That said, and as stated in definition 1, protocol correctness is guaranteed.

From table 4.1, the links where the $\otimes$ operation fails for the strictly monotone and the increasing properties are also shown. It reveals that strictly monotonicity property fails in 45 of the times and that the increasing property fails in 11 times (note, however, that this information is not visible in the table). For a better understanding of the table we consider the following example using the penultimate row (x = *d*, y = *e* and k = *c*): if the correct substitution is made we obtain $d \prec e \Rightarrow c \otimes d \prec c \otimes e$ for the strictly monotone property and $c \prec a \otimes c$ for the increasing property. By analyzing table 3.5 it is easy to see that these expressions are not met, and therefore the verification of the conditions fails (meaning that the protocol does not show both the strictly monotone and the strictly increasing properties).

Let us now use the protocol presented in section 3.5.1 with the $\otimes$ operation in table 3.1. Table 4.2 shows the Software results. By using such protocol, the program assert its non-decreasing property however, the correct behavior it is not guaranteed. From table 4.2 it is easy to see that the problematic sets L_u are: $S_L$and U_W. This results are therefore coherent with the ones obtained by hand in section 3.5.3.

Several other experiments were made in order to test the software. All the models presented so far, especially the ones introduced in chapter 3, were tested and the outcome corresponds to the theoretical results.

Protocol properties				Protocol conditions	
Strictly Monotone	Simple Monotone	Strictly Increasing	Non-Decreasing	Finite Time Convergence	Forwarding Loops
$x \prec y \Rightarrow k \otimes x \prec k \otimes y$		$y \prec x \otimes y$		Condition 3	Condition 7

The properties/conditions fails in:

x	y	k	x	y	Set L_u		Label Pair
$\overline{1}$	S_L	S_L	$\overline{1}$	D_W	U_W	S_L	$[S_L\text{-}S_L]$
$\overline{1}$	D_W	S_L	D_W	S_L			
D_W	S_L	S_L	$\overline{1}$	U_W			
D_W	U_W	U_W	S_L	U_W		U_W	$[S_L\text{-}S_L]$ and $[D_W\text{-}U_W]$
S_L	U_W	U_W	D_W	D_W			
...	...	...	...	...			

Table 4.2: Protocol program verification, from Table 3.1 (note, for the non-strict expressions $\prec$ must be changed by $\preccurlyeq$).

4.2.3 Link weight assignment

In this section the functionality for automatic policy assignment to network graphs links is described. Note, however, that manual link attribution is also possible by giving the adjacency matrix A with the link labels. The intention of this feature is to provide a way to label links based on the relative centrality characteristics of the connected nodes (as explained in section 4.2.1).

The implemented program functionality uses a hierarchical method to assign the elements of the protocol set *S* to links. This is done by using the graph characteristics in conjunction with the relative preference of set *S* elements. First, the program chooses one of the graph centrality characteristics: node degree; betweenness; or eigenvector, to be its deciding factor, which is done by choosing the characteristic with more differentiated values for a particular graph. The next step, is to lookup which node has the highest centrality value (which will be the starting point of the

labeling algorithm). When this node is selected the program starts its link label assignment operation by realizing the following operations:

1. Search the centrality values for neighbors nodes;
2. For every link of those neighbors the assignment will be: if the centrality value is greater then, one of the worst policies of the set is assigned; if the centrality value is equal (or very similar), the link will be labeled with one of the middle policies presented in the set; and if the centrality value is worst, the link assignment will be performed with one of the best preferred policies in the set;
3. Next, the policy in the reverse direction of the link is assigned according to the label pair definition of the protocol model;
4. The last step is to choose the neighbor with the highest centrality value to be the main node for the next iteration. If a node was chosen a new iteration starts returning to operation 1, otherwise the program ends (if there is no more nodes, it means that this process has been performed in all nodes).

At the end of this program cycle, a network topology adjacency matrix *A* with links labeled with elements from the *S* set of the protocol model is obtained. The major limitation of this algorithm is when almost every nodes in the network graph have similar centrality values, leading to a link labeling based on the middle elements of the order preference. The consequence is that too many bad circuit labelings can occur. To overcome this kind of limitations a simple and basic solution was implemented, it consists on only using the best and worst elements present in the *S* set (apart from trivial and invalid paths, respectively $\overline{1}$ and $\overline{0}$). In this way non-free cycles are excluded although policy diversity is lost.

The results of this automatic labeling are limited and can only be used in certain graphs. The knowledge of the graphs characteristics can be used to predict its usefulness. In our case we are interested in Telecommunication type networks. Such networks can be represented by graphs with very distinct characteristics.

There are a few main categories of graphs to consider. Random graphs [44], are graphs where the connectivity distribution of a network peaks at an average value and decays exponentially and are also called “exponential networks” or “homogeneous networks,” because each node has about the same number of link connections. Scale-free graphs [45], in these graphs connectivity distributions are in a power law form that is independent of the network scale. Differing from an exponential network, a scale free network is inhomogeneous in nature: most nodes have very few link connections and yet a few nodes have many connections (with the Internet being one of the most notorious cases). And finally hierarchical networks where connectivity depends on the nodes position in the hierarchy.

In figure 4.1 a comparison between typical networks topologies and their degree distributions is presented: in left a characteristic function of random networks (A); in center a common function of a scale-free network (B); and right a hierarchical network (C).

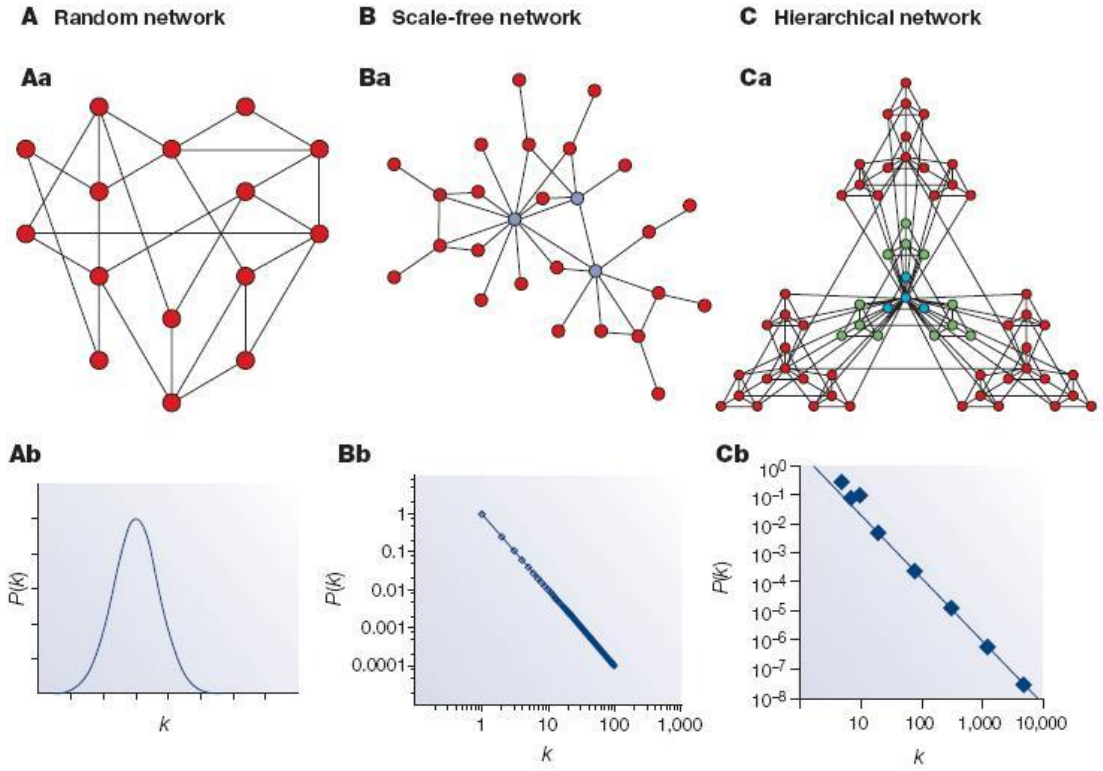


Figure 4.1: Degree distribution of an exponential (A) and two scale-free networks (B and C) [48].

From figure 4.1 we see that random networks have an exponential degree distribution, meaning that almost every nodes present the same node degree, which therefore leads to similar centrality values for the majority of the network nodes. Under this circumstances we fall into the problem mentioned above, the difficulty to attribute link weights automatically without compromising finite time convergence. In the other two networks illustrated in figure 4.1 we can use the automatic link weight assignment. Random networks, however are not well suited for the algorithm used in the tool.

4.2.4 Path calculation Algorithm

This section presents the matrix iteration algorithm to calculate the routing solution and the algorithm used to measure path diversity in that solution.

4.2.4.1 Matrix iteration

The matrix iteration algorithm is quite simple and its theoretical explanation can be found in section 3.5.2. This section explains how it was implemented.

First, we introduce the data structures for the algorithm, as explained in section 4.2.2. The set elements are expressed in our implementation as integer values, so we match integer values to the sets *S* following the preferred order in the set (i.e. eq. 3.9).

The first operation is for path selection corresponding to the algebraic $\oplus$ operation, which selects the most preferred given two elements of set *S*. The second operation is the path composition operation, which calculates the resulting path from extending two paths, as showed in 3.5.2. Using the algebraic $\otimes$ operation for set *S* (as the one in table 3.5) we obtain the paths weight, *w*(*P*).

From this we build the adjacency matrix to represent a network topology, i.e. each position of the matrix stores a policy (an element of set *S*) which is the path value for an origin-destination pair. We represent the network matrix as *A* [*X*][*Y*], where the lines are the origins [*X*] and columns correspond to each destination [*Y*]. Note that, the function that populates each position of the matrix also deals with the reverse link policy according to the label pairs (definition 16).

The function to multiply matrices used for the matrix iteration algorithm is similar to the ordinary matrix multiplication the difference being that instead of the multiplication operation the path composition operation is applied and the sum is replaced by the path selection operation. In the path selection operation more than one node can be preferred as next hops so each position of the matrix has to have also a list of next hops (in order to be used for multipath forwarding). Table 4.3 illustrates the data structure used for each matrix position.

Source	Destination	Path weight (w(P))	Set of next hops H_y
X_i	Y_j	w(P_y) ∈S	$H_y = \{X_1, X_2, ..., X_n\}$

Table 4.3: Adjacency matrix, *A*, data cell structure.

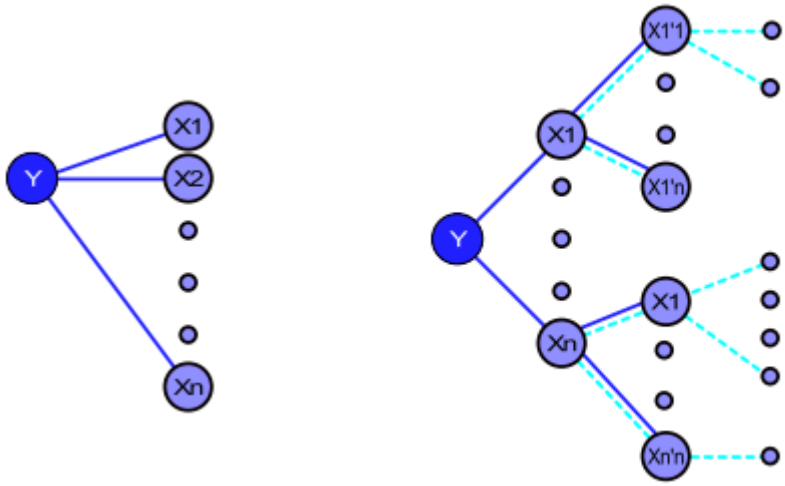
The input for the matrix iteration algorithm is a topology, which contains all connections between neighbors, and the policies applied in those connections. The topology is given through a text file (in a similar process to the protocol model) where each neighbor pair has a policy value associated to the connection. This data is loaded into the adjacency matrix *A*. The matrix iteration algorithm obtains the routing solution *A**, by left multiplying matrix *A* recursively (eq. 3.6) until condition (eq. 3.7) is met.

4.2.4.2 Path discovery

After calculating a routing solution using the matrix iteration algorithm, it is possible to generate the forwarding tables by extracting information from *A**. Note that each line of the matrix corresponds to the forwarding table of one of the nodes. Because matrix *A** contains a set of next hops to each reachable destination, it is possible to identify the existing paths between each pair of nodes.The path discovery algorithm explores all forwarding possibilities (for each source-destination pair), starting from the destination and builds all the possible paths to each source. If we take a column from matrix *A**, we get the next hops from all sources to a destination.

To calculate all paths from a destination to all sources, first we simplify the search by organizing the information. Instead of having a source with a list of next hops like a forwarding table we change it to a next hop with a list of sources, i.e. a list of sources that possess the index node as next hop.

The idea is to first add all directly connected hops to the destination, from the simplified search list. These links are also paths from those hops to the destination, as illustrated in figure 4.2 (a). Now by adding to these hops their next hops, shown in figure 4.2 (b), we can get both paths of sources at two hops distance from the destination and/or alternative paths of sources at one hop distance. By appending hops recursively we get all paths of all sources to the destination. We then run the path discovery algorithm on every destination, to find all topology paths.



(a) First hops *(b) Second and third hops*

Figure 4.2: A scheme representing the path discovery algorithm.

After all possible paths between each pair of nodes in the topology have been found, they can be classified into two categories:

- Disjoint paths - paths that do not share any link or node other than the source and destination;

- Different paths - paths that share at least one link or node but with at least one different.

It is important to distinguish the multiple routes since the goal is to have as many disjoint paths as possible.

In protocols with destination-based and hop-by-hop forwarding, the actual path followed by packets will depend on how each node distributes traffic between the possible next hops it has in the forwarding table.

4.3 Simulation results

We performed a set of experiments with different network topologies using the protocol models verified to work correctly using our software. These experiments are described in the next subsections.

In the first set of experiments, presented in Section 4.3.1, the network graph has a regular topology. For that reason we manually assigned link weights (with a hierarchical structure labeling in mind).

The protocol model is the one presented in section 3.6.1 with the $\otimes$ operation in table 3.5 (since theoretically it seems to be the most generic possible protocol, as described in Section 3.6.3). However, and to become easier to understand, we will assign a more specific meaning to the elements of set *S*.

The protocol model is intended for use in networks with hierarchical structure and meshes at the various levels connecting sets of nodes. Intuitively the model uses preferentially the lowest possible mesh to go to the destination (with multipath) and rises in the hierarchy until a horizontal solution is possible (also with multipath). Therefore, hierarchical (vertical) multipath is performed by having several “local roots” on the network hierarchy. Horizontal multipath is obtained by having several paths to the “right”, or “left”. The protocol model is as follows.

The set *S* is represented by:

$$S = \{\overline{1}, D, S, R, L, U, \overline{0}\} \qquad \text{(Eq. 4.1)}$$

note that, each element corresponds, respectively, to the set $S = \{a, b, c, d, e, f, g\}$ used in table 3.5. The set order preference is:

$$\overline{1} \preccurlyeq D \preccurlyeq S \preccurlyeq R \preccurlyeq L \preccurlyeq U \preccurlyeq \overline{0} \qquad \text{(Eq. 4.2)}$$

the label pairs considered are:

$$[\overline{1}\text{-}\overline{1}] \wedge [D\text{-}U] \wedge [S\text{-}S] \wedge [R\text{-}L] \wedge [\overline{0}\text{-}\overline{0}] \qquad \text{(Eq. 4.3)}$$

and the $\otimes$ operation is:

$\otimes$	$\overline{1}$	D	S	R	L	U
$\overline{1}$	$\overline{1}$	D	S	R	L	U
D	D	D	$\overline{0}$	$\overline{0}$	$\overline{0}$	$\overline{0}$
S	S	S	$\overline{0}$	$\overline{0}$	$\overline{0}$	$\overline{0}$
R	R	R	R	R	$\overline{0}$	$\overline{0}$
L	L	L	L	L	L	$\overline{0}$
U	U	U	U	U	U	U

Table 4.4: Protocol model $\otimes$ operation 3 (for experiments).

Table 4.4 assures the correct operation of the protocol since it is in concordance with the one presented in section 3.6.3 (note, that only the name of the set elements are changed). We also verified it with our software that shows the non-decreasing property of the model and that it is within the finite time convergence and forwarding loop freeness conditions (conditions 3 and 7).

4.3.1 Regular topologies

The first topologies used for the simulations are highly regular and hierarchical. They are composed by an accesses/edge level, aggregation level and a core level, which is a common network topology design that a Data Centre, University Campus, Enterprise Campus or Service Provider could use. The hierarchy with redundant connections between levels achieves: robustness, high availability and throughput.

The first experiment uses a common fat-tree topology whereas the second experiment use a typical service provider aggregation network.

4.3.1.1 Fat-tree topology experiments

In this first set of experiments a typical fat-tree topology is used to analyze the impact of having more or less links, with different labelings, in the path diversity of the routing solution. The intention is to understand under which conditions a typical fat-tree can offer a better multipath solution.

The first experiment is a simple fat-tree with few links between nodes at the same level. In the second experiment we will see why the higher levels of a hierarchical topology must have as much links as possible between them (thus offering a greater degree of multipath solutions) and in the third experiment a different link labeling solution to the same topology is provided. The last experiment of this section consists on applying the automatic link weight assignment to the topology. Finally, and before concluding this set of experiments, a comparison between them is made.

First experiment

In the first experiment we use figure 4.3 as the network graph model. The topology contains 28 nodes and 52 links divided in three hierarchical levels. Blue, green and yellow colors represent levels 0 (edge), 1 (aggregation) and 2 (core) respectively. We have 16 nodes at level 0, each one of them connected to two nodes at level 1 so that we have at least two paths towards level 1 in those nodes. At level 1 there are 4 clusters of aggregation formed by 2 nodes each, making the total of 8 nodes at level 1. There are 4 nodes at level 2 that serve as the core.

Links between nodes in different levels are labeled with *U* or *D* labels depending on the direction. Same level link *S* labels are used between nodes in the same level of the hierarchy. Each level 1 node is connected to two level 2 nodes via *U*/*D* links.

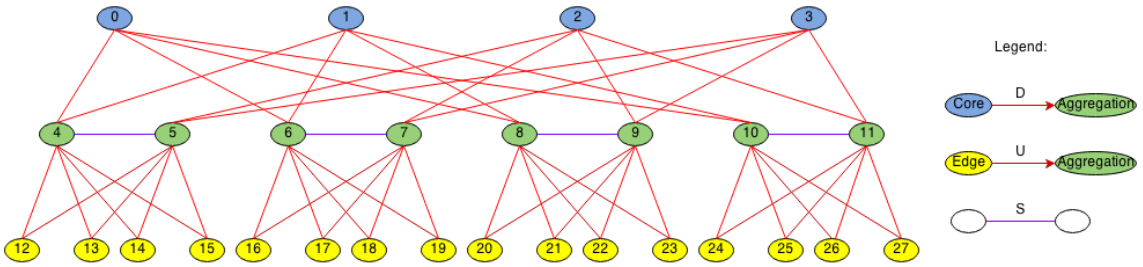


Figure 4.3: Fat-tree topology (first experiment).

We calculated the routing solution for the topology with the described labeling of the links and then measured the amount of different paths for all source-destination pairs of the topology. We obtained two forwarding entries for 51.9% of node pairs, one forwarding entry for 39.2% and no forwarding for 8.9%. An important observation made is that every source-destination pairs within the edge level have at least one path linking them, and the 8.9% of pairs with no entries are located both in the core and the aggregation level. The lack of paths between some nodes is strictly related to the protocol model since its design is intended for use in networks with meshes at various levels, principally at the “higher” levels (which is not the case for this topology).

In a more concrete analyze, the null paths mentioned above are related to invalid entries in table 4.4. Take the examples of the source-destination pair, node 0 and node 5, in both directions. From node 0 to node 5 the null path is related to the entry $D \otimes S = \overline{0}$ and inversely, from node 5 to node 0, to the $S \otimes U = \overline{0}$. Take as notice that these two cases are associated to the need to reduce the L_u sets so that forwarding correctness is achieve for all possible labelings, in order to respect condition 7. Considering the first case, and looking to table 3.4, we see that four possible non-null results could be used for that $\otimes$ operation (*S*, *R*, *L,* and *U*), since for all of them condition 7 is met. However, for the second case the solution is not so simple, because in order to maintain the protocol model structure with a non-decreasing operation the only possible path is indeed the null path (accordingly to table 3.4). A different situation leading to a null path in the topology is related to paths that cannot rise, again, in the hierarchy after going down, to lower levels (observed among the core nodes). An example of such situation is the null path between the source-destination pair, nodes 0 and 1. The operation in question is: $D \otimes U = \overline{0}$, and a possible alternative solution is substituting the result to a *U* path weight and therefore substituting the result of $U \otimes U = \overline{0}$ (as stated in section 3.6.2 and also showed in table 3.4). By performing such operations the protocol model still meets the necessarily conditions, conditions 3 and 7, for its correct behavior.

The resolution for the lack of connectivity consists in modifying the network topology, where two different approaches can be applied, each one providing a different solution:

– by creating links between nodes in the core level, which solves all the null paths between source-destination pairs in all the three levels (since the core is the highest level);
– or by connecting the missing links in the aggregation level, solving the null paths presented in the two lowest levels (the aggregation and edge levels).

The simple above examples show how important the formal study of the protocol can be and how our software can be useful. Note, that this is also significant in order to make informed decision to augment the multipath degree.

Second experiment

The second experiment presents one of the solutions provided in the first experiment. The solution adopted is to link the core nodes and see if the problem of null paths still exists and if the number of paths is better. Since the core nodes are all in the core level, the labeling is made by using the element/policy *S* of set *S* for that specific purpose. The new fat-tree topology is illustrated in figure 4.4.

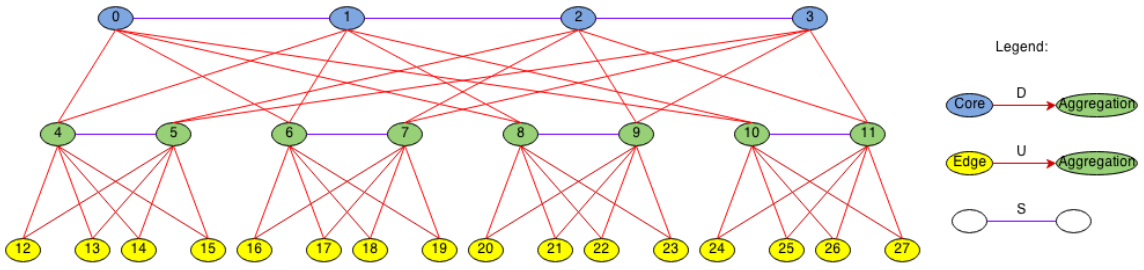


Figure 4.4: Fat-tree topology (second experiment).

Like in the previous experiment we calculated the routing solution for the topology with the described link labeling and then measured the amount of different paths for all source-destination pairs of the topology. We now obtained two forwarding entries for 70.9%, one forwarding entry for 26.2% and no forwarding for 2.9%.

Comparatively to the first experiment we verified a considerable increase of the forwarding entries. Although there is an increase in the number of paths, we find out that some of source-destination pairs still do not have a possible path connecting them. The reason for these missing paths is related to the fact that consecutive links of policy *S* need to be invalidated to prevent the appearance of forwarding loops (in order to meet condition 7).

So, to overcome this model limitation we can use two different labels *R* and *L* instead of the *S* label in the same level links. We explore this possibility in the next experience.

Third experiment

The *S* label is changed by two labels *R* and *L* that are a label pair. This replaces the label pair [*S*-*S*] for [*R*-*L*] this makes the algebra compliant with condition 7. The changes made to the topology of figure 4.4 are illustrated in figure 4.5.

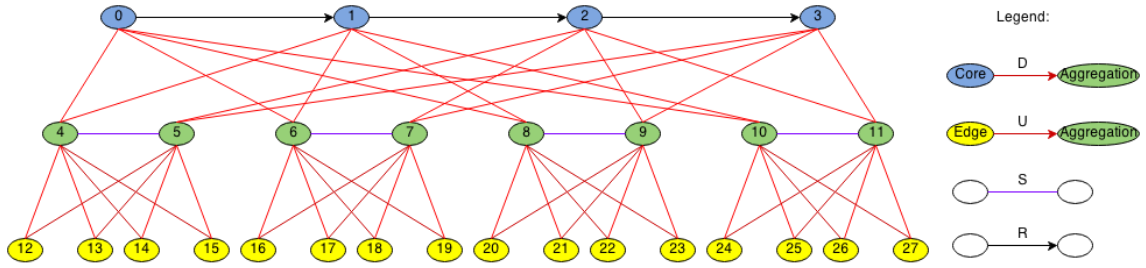


Figure 4.5: Fat-tree topology (third and fourth experiments).

Similarly to the previous experiments we calculated the amount of different paths for all source-destination pairs of the topology. We now obtain two forwarding entries for 80.42% and one forwarding entry for 19.58%, meaning for all source-destination pairs there is at least a path connecting them. The 19.58% of pairs with a single entry are mainly pairs between the core nodes (0, 1, 2 and 3) and the result is related with the lack of a full mesh between them. For the rest of the pairs, either in the aggregation and the edge levels, there are in general two forwarding next hops in the forwarding table.

The existent paths between each pair of nodes were calculated using matrix *A**. We then classified them into disjoint paths and different paths. The graph of figure 4.6 shows the cumulative distribution for the number of paths per source-destination pair considering both disjoint and different paths.

The ideal curve would remain low, and increase at the end (meaning that a large percentage of nodes have high path diversity). In the topology there are, at most, two disjoint paths between any pair of nodes. We can see in the figure 4.6 that two is in fact the limit in the number of disjoint paths. However, we see that 55.2% of the source-destination pairs only have one disjoint path and 44.8% have two disjoint paths between them. Considering the different paths we see that 23.3% of the pairs have a single path. This is related with the cases above where only one entry exists in the forwarding table. They are due to the lack of diversity in the topology (between nodes in the core) and the design choices of the preference order that prefers labels *R*, *L* to *U* with *R*, *L* being used in our labeling in the links of the closest path at the same level instead of the up direction that provides more links.

The path diversity is slightly higher in this case, with the highest number of different paths between some pairs reaching the value of 4. As seen in the figure 4.6, 44.7% of the pairs have three or less paths and the others 55.3% have four or less different paths.

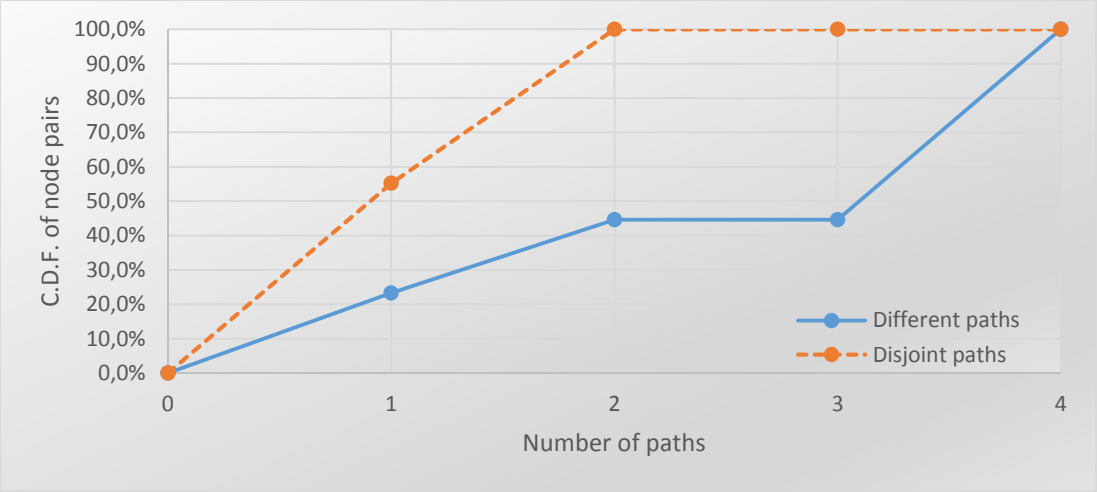


Figure 4.6: C.D.F. of the number of different paths per source (fat-tree 3).

This simple experiment illustrates that, using different link labelings, there are different numbers of paths. From the experiment, we see that maintaining the simplicity of simple destination based hop-by-hop forwarding, and using a correct link labeling, the path diversity degree may increase.

Fourth experiment

The fourth and last experiment using a fat-tree topology make use of the software automatic link weight attribution functionality. Note that, the topology used is an abstraction of the fat-tree network topology presented in figures 4.4 and 4.5. This is related to the fact that the program only have to know each nodes are linked. The weight of the links are not considered since they are obtained automatically through the execution of the program. The centrality value choose was the degree centrality, since whenever possible (different centrality values per nodes) the software uses this centrality measure as its first option.

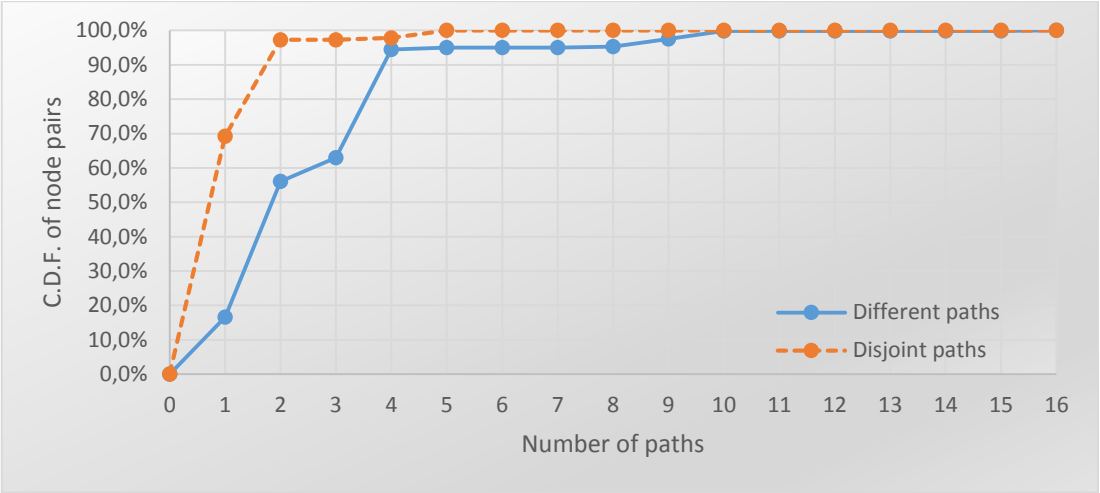


Figure 4.7: C.D.F. of the number of different paths per source (fat-tree 4).

The solution obtained shows that one forwarding entry exists for 32.8%, two forwarding entries for 59.8%, four forwarding entries for 0.2% and five forwarding entries for 7.2%. By analyze figure 4.7, which represent the cumulative distribution for the number of paths per source-destination pair for both different and disjoint paths, we see that the maximum number of different paths is sixteen (which only occurs between one pair of nodes). The pair case with sixteen different paths only represents 0.1% of the different paths presented in the topology and the majority of the pairs contains four different paths, representing 94.4%. Observing the disjoint paths per source-destination pairs we see that one disjoint path is represented by 69.2% of the total amount of disjoint paths, 28.1% with two, 0.5% with four and rest 2.2% with five.

Experiments comparison

The first and second experiments presented exhibit some node pairs that are have no paths and the algebra and labelings of the third experiment solve this. They take into account condition 7, and therefore we will consider the third experiment as the best between these first three experiments.

Although the third and fourth experiments have different labelings, they both implement the same protocol model to the same topology. Therefore we compare their results in the following table.

Path	3º Exp.	4º Exp.	[%]
Different	1828	2048	12.0
Disjoint	388	256	-34.0

Table 4.5: Path diversity between the third and fourth experiments (fat-tree topology).

From table 4.5 it is easy to see that the total number of different paths, from the third to fourth experiments, do not differ much. However, for the disjoint paths case a considerable decrease is verified. From previous experiences we see that connecting the core nodes provides a great improvement to the number of possible paths in the topology, including different and disjoint paths. Since we define, in the protocol model, the preferred path by going down *D* in the hierarchy it becomes easy to understand that nodes in the core level (the highest level in the topology) will contain the best paths to all destination in the network topology. This means in a hierarchical structure the more nodes the higher level contains, the more path diversity it will exist. Therefore, the more number of links connecting different levels, the more number of disjoint paths is present (which is great for a multipath routing solution).

In our opinion, for regular networks and using this kind of protocols model, the automatic link attribution offered by the software provide a quick manner to label the links and taking centrality measures into account. Giving preference to the less central nodes corresponds to a similar result to the manual labeling we performed.

4.3.1.2 Typical provider aggregation topology experiment

In this second set of experiments, we will study the properties of the multipath protocol model under a typical provider aggregation topology. This type of topology, like fat-tree networks mentioned before, also present a hierarchical structure. However two major differences are present: the core level presents a full mesh aggregation and the aggregation level is also fully connected.

The experiment is divided in two different approaches, both using the same protocol model and topology. The first experiment uses the core level as the highest level of the topology, meaning the policies/weight applied to links result in paths that prefer not to rise in the hierarchy, or to rise the less possible. This is a characteristic of the protocol model, since by the $\otimes$ operation, defined in table 4.4, a path only rise in the topology hierarchy until a possible horizontal solution appears. In the second experiment, an inverse topology hierarchy is applied (we will invert the link policy with

their respectively pair link). Or, in other words, the core level becomes the more preferred level in the hierarchy and the edge level the least.

First experiment

For the first experiment we will use figure 4.8 as our network graph. We can see that the topology contains 28 nodes and 66 links divided in three hierarchical levels. Blue, green and yellow colors represent levels 0 (edge), 1 (aggregation) and 2 (core) respectively. We have 12 nodes at level 0, each one of them connected to two nodes at level 1 so that we have at least two paths towards level 1 in those nodes. At level 1 there are 4 clusters of aggregation formed by 3 nodes each, making the total of 12 nodes at level 1.

Links between nodes in different levels are labelled with U or D labels depending on the direction. Same level link labels are used between nodes in the same level of the hierarchy. Within a cluster the nodes are connected via *R* or *L* links so that local horizontal traffic is possible (and condition 7 is assured). *S* links are used between the 4 clusters meaning that horizontal traffic should only occur within the cluster or at most to the neighboring clusters. Finally each level 1 node is connected to two level 2 nodes via *U*/*D* links.

There are 4 nodes at level 2 that serve as the core. They are connected via *R*/*L* links. Due to this structure, the highest path diversity expected in the topology is when a node at level 0 tries to reach another level 0 node that is not in its neighbor cluster. The protocol states that the link weight *S* is used to connect nodes at the same level. However, and as stated so far, symmetric links (or label pairs as this dissertation usually calls them) cannot be tied together (from table 4.4, $S \otimes S = \overline{0}$), thus limiting the possible existing paths. So, in order to overcome this limitation we use the *R*/*L* elements of the protocol model.

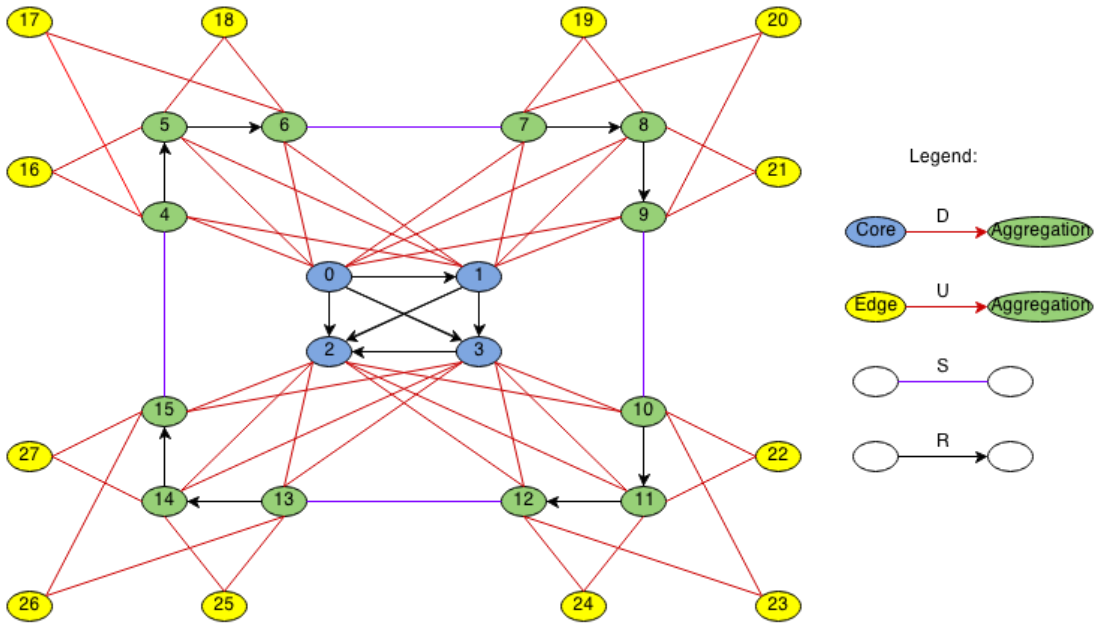


Figure 4.8: Typical provider aggregation network.

One example of such cases is illustrated between nodes of level 2 and will serve as an example for the correct labelling in this kind of situations. The first one is (0, 1, 3, 2, 0) going around this cycle in that direction has at least one differently labelled link due to the link 2-0 being *L* and all the others *R*. If we circle in the other direction, then all links are *L* but 0-2 is *R* for this direction. The same observation can be made for the crossover links. Note, the labeling is performed in such way to exclude the possibility of having a non-free cycle (definition 10) in the network to assure finite time convergence.

We calculated the routing solution for the topology with the described labelling of the links and then measured the amount of different paths for all source-destination pairs of the topology. We obtained three forwarding entries for 3.7% of the source-destination pairs, two forwarding entries for 75.1% and one forwarding entry for 21.2%. The topology is designed so that for almost every destination, each node has at least two forwarding options. It was then expected that the majority of the pairs had two different entries in the forwarding table.

The graph of figure 4.9 shows the cumulative distribution for the number of paths per source-destination pair considering both disjoint and different paths.

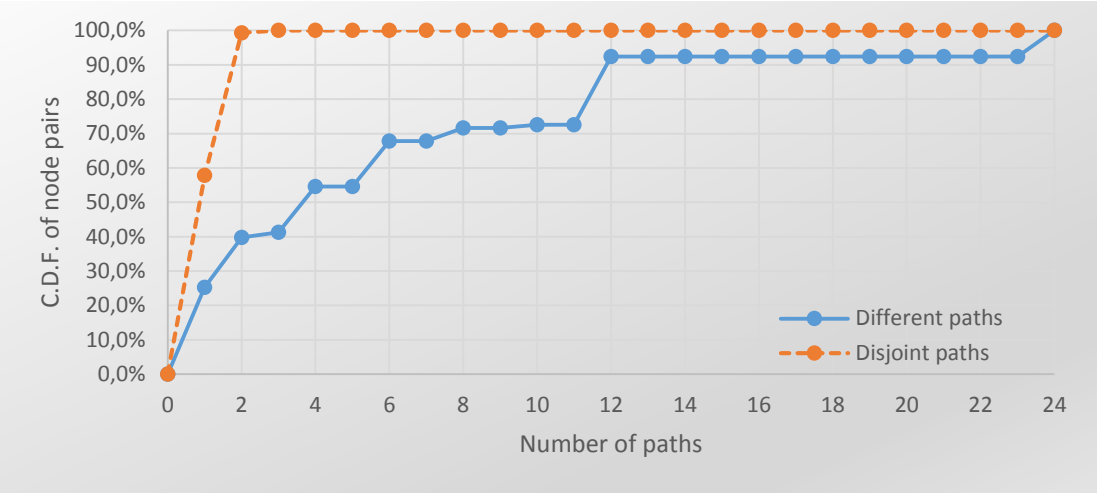


Figure 4.9: C.D.F. of the number of different paths per source (provider aggregation 1).

Remember that the ideal curve would remain low, and increase at the end (meaning that a large percentage of nodes have high path diversity). From figure 4.9 we see that two is the limit number of disjoint paths. However, we see that 57.8% of the source-destination pairs only have one disjoint path, 41.5% have two and 0.7% have three disjoint paths between them. Considering the different paths we see that 25.2% of the pairs only have a single path, which is strictly related to the cases where only one entry exist to the forwarding table. They are essentially due to the design choices at certain parts of the network that prefer the shortest path to the destination instead of the direction that provides more links. Path diversity is higher in this case, with the highest number of different paths between some pairs reaching the value of 24. As seen in the figure 4.9, 54.6% of the pairs have only three or less paths, 32.2% have six or more possible paths, and 7.6% have twelve or more different paths.

This simple example of a common hierarchical network illustrates that, compared to the fat-tree experiment, path diversity can be considerable augmented when using a full mesh between core nodes (and when using a protocol model as ours).

Second experiment

The first experiment realized with the provider aggregation topology uses a hierarchy, defined by the link policies assigned by us, where the core is highest level and therefore the least preferred. The second experiment, however, uses a different approach to the topology hierarchy. In this experiment we will define the lowest level as being the least preferred level (level 0), the aggregation level as the middle level (level 1) and the core level as the more preferred level in the network topology.

The inverted hierarchy for topology of figure 4.8 is performed by a function in the software. The function realizes a very simple task, which is no more than substituting the policies of the links (presented in the adjacency matrix *A**) by their respectively pair link. The result is the topology illustrated in figure 4.8 but with a new legend, showed in figure 4.10.

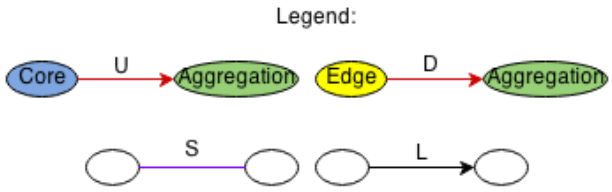


Figure 4.10: Service provider topology (figure 4.8) legend for second experiment.

The topology used in this experiment, along with our protocol model, presents some different characteristics to the ones presented in the first experiment. In this experiment, paths prefer to go to the core. This is simply because paths from higher nodes will have a weight/policy of *D* to reach the core level nodes, remember that this policy is the preferred element of set *S*. However, in the other hand, paths leaving the core will have the worst policy possible (accordingly to the model), since to reach the other levels they have to rise in the hierarchy and therefore having a path with weight *U*. In order to better understand the impact of such topology link label change, we will perform the same type of analyze as for the first experiment.

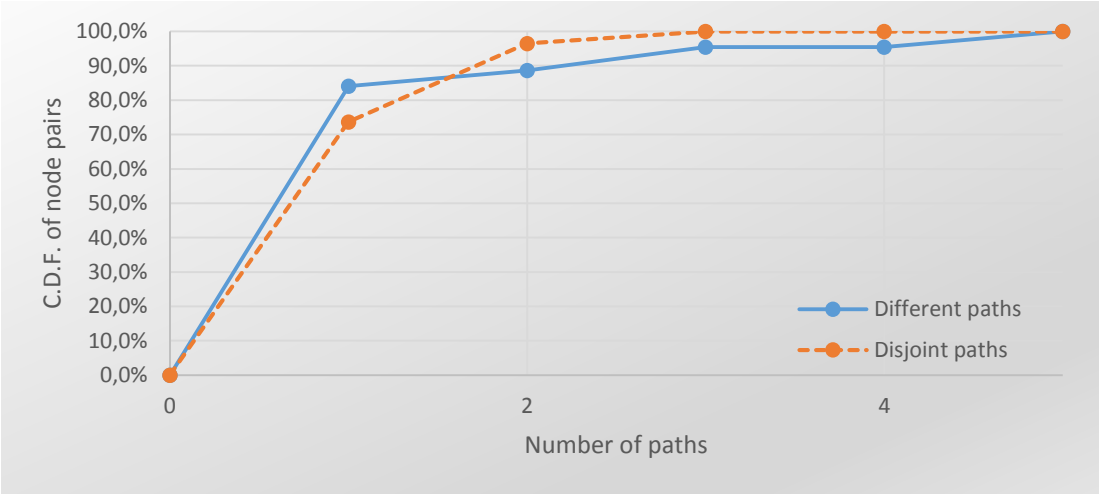


Figure 4.11: C.D.F. of the number of different paths per source (provider aggregation 2).

By using the topology of figure 4.8 with the described labelling of links we calculate the routing solution. A first important observation is the appearance of many null paths between source-destination nodes, especially in edge nodes. The absence of paths observed in the forwarding entry table represents 68.8% of the total amount of node pairs. Another important observation is

that the majority of nodes with more than one forwarding entry are located in the core level. In fact we see that in 1.1% of the cases there are six forwarding entries, and all of them corresponding to paths starting in the core nodes.

The graph of figure 4.11 considers all nodes in the topology and shows the cumulative distribution for the number of different and disjoint paths per source-destination pairs. We see that three is the maximum number of disjoint paths, which corresponds to eight pairs with three disjoint path. They are related to paths from core nodes to aggregation nodes. The source-destination pairs with only one disjoint path represent 73.7%, which is far from ideal. By analyzing the different paths we see that 84.1% of the pairs only have a single path, 7.1% have three and 4.5% have five.

Experiments comparison

Table 4.6 compares both scenarios to see the impact of the changes made to the labelings. As observed the number of paths between source-destination pairs fell abruptly from the first to second experiments. Therefore, a decrease in the number of different and disjoint paths also occurs, as table 4.6 reveals.

Path	1º Exp.	2º Exp.	[%]
Different	4130	240	-94.2
Disjoint	420	296	-29.5

Table 4.6: Path diversity between the first and second experiments (provider aggregation topology).

From table 4.6 we verify that the principal decrease happens in the number of different paths. The result is quite obvious since in the second experiment a lot of null paths appeared. However, we see that the number of disjoint paths do not suffer that tremendous decrease. This fact is also related to the great number of null paths. The reason is the algebra operator $\otimes$, of the protocol model results many times in an invalid path in the second case (strictly related to the hierarchy definition of the protocol).

The set of experiments realized allows us to understand the importance of the protocol model being adapted to the topology in where it is applied by a correct link labelling of the network topology. If this consideration is not taken into account when planning the routing in a network then path diversity can be in question. So, in order to have a well-design multipath routing protocol operating under a regular topology, a careful link labelling, with a special attention to the hierarchical structure, is a must.

4.3.2 Power-law topology

A power law degree distribution graph, is a completely different type of network. It has a much looser hierarchy. The Internet inter AS networks is an example of a power law graph, in the Internet case a certain hierarchy is induced by the business relationships between the ASes, but in this case there is not a clean topologic structure with clear levels like in the previous topologies. There are links directly connecting lower level nodes with higher level nodes (without passing through intermediate levels) and also same level links between different levels of the hierarchy. In order to understand the behavior of a multipath policy based protocol in such a network we used a small power-law topology. The topology, illustrated in figure 21, was obtained from the graph generator, NetworkX, and has 56 nodes (which can be abstracted as ASes). Note, a bigger topology could have been used, however excessively computer resources would be consumed.

The Internet inter AS network is closely modelled by a long-tail distribution of node degree. This means that a small number of nodes has high connectivity and a large number of ASes has a small number of connections (a characteristic of power law topologies). The graph of figure 4.13 shows the number of connections per node (node degree) as well as the cumulative distribution function (C.D.F).

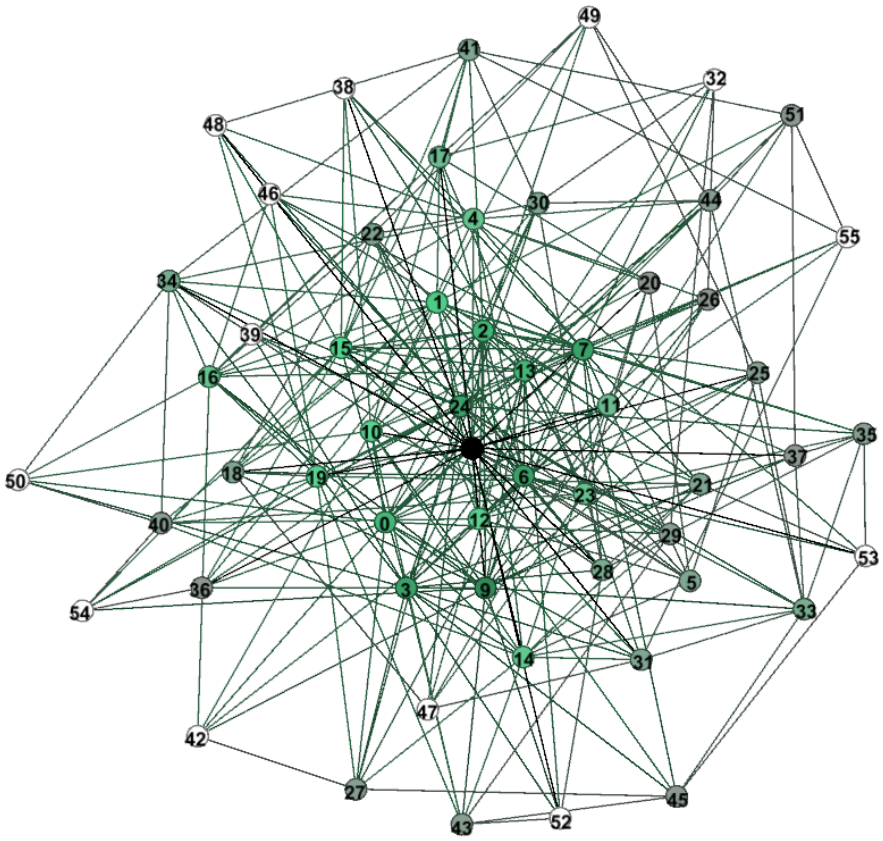


Figure 4.12: Scale-free topology.

We can see in figure 4.13 that the degree distribution of the topology of figure 4.12 has indeed a power-law distribution.

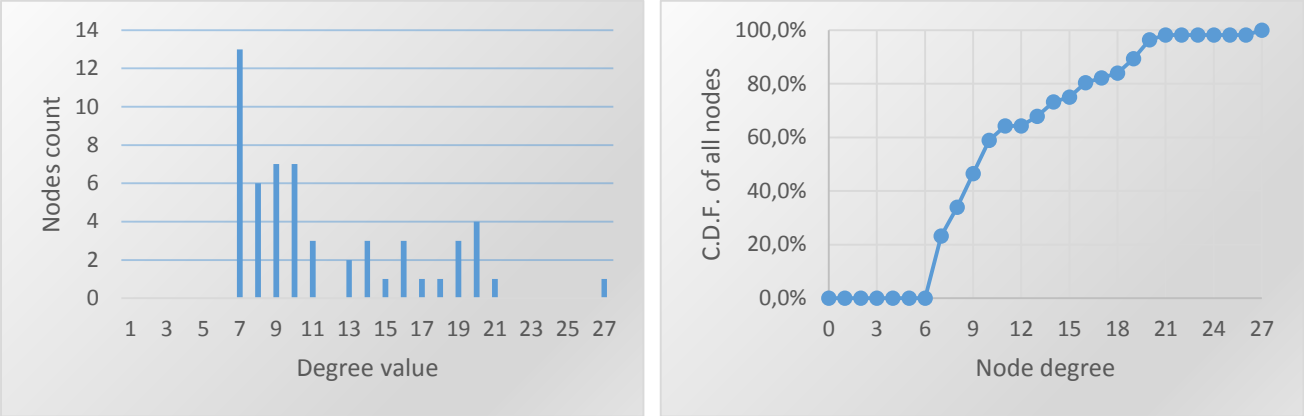


Figure 4.13: Degree distribution.

4.3.2.1 Scale-free topology experiment

For this topology we used the same protocol model as before (presented in the beginning of the section), and performed two different link labelings for the topology. The assignment was made automatically since the topology has a power law degree distribution and manually deciding labelings for the entire topology is time-consuming.

In the first experiment we only used the preferred and worst element of set S, the label pair [*D-U*], for the topology link labelling (remember that for our software the elements $\overline{0}$ and $\overline{1}$ have always to be present). Therefore, the topology does not have nodes considered of the same level and connected by the intermediate preference labels. For the second experiment we used all elements in set *S*. In this later scenario, whenever there are two consecutive *S* links connected, one of the two policies *R* or *L* is applied.

The graph of figure 4.14 considers all nodes in the topology and shows the cumulative distribution for the number of different paths per source-destination pairs in the first and second experiments.

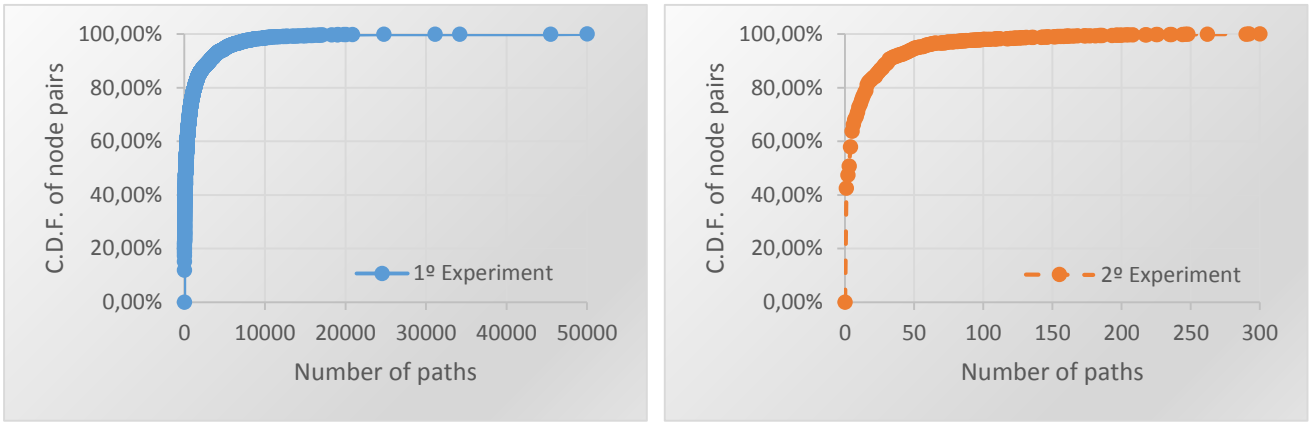


Figure 4.14: C.D.F. of the number of different paths per source (scale-free 1 and 2).

The distribution comparison was divided in two distinct graphs, because as we can see there is a great discrepancy between the two experiments. In the first experiment, at least, a pair of nodes contain 50000 different paths whereas for the second experiment the maximum number of paths between a pair source-destination is 300. We can see a great decrease in the number of different paths each node has for the second experiment. In fact table 4.7 shows that number of different paths decreased 98.0% compared to the first experiment.

Path	1º Exp.	2º Exp.	[%]
Different	3262450	50357	-98.0
Disjoint	1045	2367	126.5

Table 4.7: Path diversity between the first and second experiments (scale-free topology).

All nodes decreased the total number of different paths to all destinations and a possible reason for this is the absence of same level nodes. By only using the label pair [*D*-*U*] policies a great number of paths have the same preference (since we only have two classes of preference). In the second experiment, nodes at same level were introducced, therefore decreasing the number of paths that can be considered equal, since the set order preference is augmented (all policies are taken into account). Having a higher diversity of policies leads to a decrease of the number of equally preferred paths.

However, and as observed in table 4.7, despite the great decrease of different paths between the first and second experiments, we see a smaller variation in the disjoint paths case. Considering

figure 4.15, where the comparison between the cumulative distribution for the number of disjoint paths per source-destination pairs of the two experiment is illustrated.

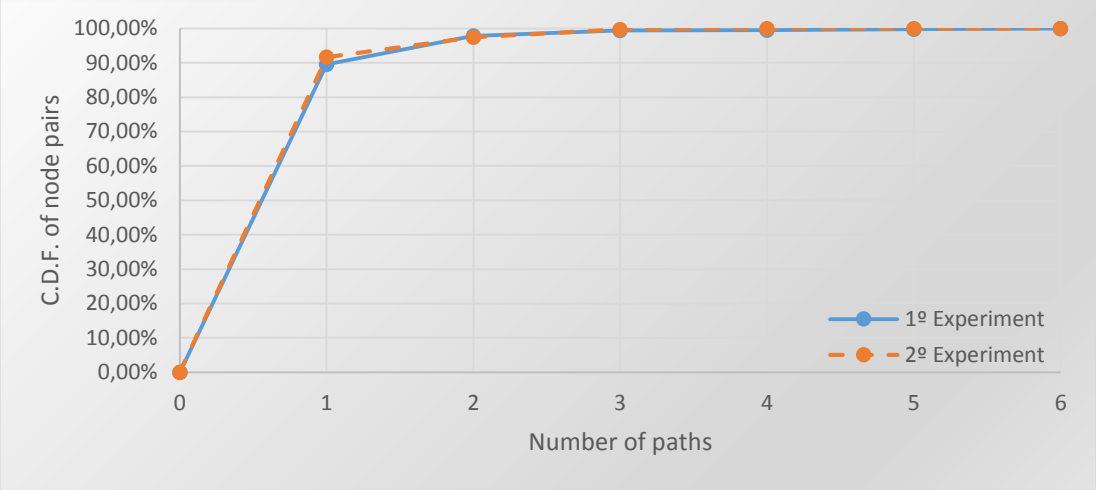


Figure 4.15: C.D.F. of the number of disjoint paths per source (scale-free 1 and 2).

From figure 4.15 we see that the disjoint paths C.D.F. for both experiments are virtually identical, the difference is only verified in the number of source-destination pairs. For example: the first experiment have 822 pairs with only one disjoint path whereas for the second experiment there are 1863 pairs with one disjoint path; and considering the number of node pairs with four disjoint paths we observed for the first experiment 4 and for the second 24. The number of nodes with only one path is much smaller for the first experiment (which is consistent with the increase of different paths). The increase of pair nodes with 4 disjoint paths is related to the densely connect nodes that were connected with *R* and *L*, therefore taking advantage of the path diversity between them.

The results obtained in the first experiment rise up the hypothesis of having much more different paths diversity with less policies. It seems that as more labels (policies) are used less paths we have since there are more equivalence classes (in terms of preference) through which the available paths can be distributed. If we consider the application in a real network like the Internet inter AS network for example the idea of peer policies (same level policies) is indeed to reduce the available paths because the aim is to have the traffic going through peers since it is cheaper.

CHAPTER 5: Conclusions

Multipath routing protocols are more complex than traditional single shortest path protocols and the conditions in which they correctly operate are more difficult to understand and define. In this thesis we isolated the main results from the existent work and provided a simple set of possible conditions that need to be met such that these protocols operate correctly. We also presented a generic multipath hierarchical routing protocol to be used in networks with distributed control and independent nodes that converges to a solution in finite time and does not have forwarding loops. Therefore, we tackle some of the current open issues in multipath routing namely the lack of correctness guarantees and the poor TE capabilities.

In order to have a multipath routing solution, by increasing the number of paths that can be considered equally good, the choice was to use a non-decreasing model with some restrictions on its $\otimes$ operation and as well as some restrictions on the network graph, therefore assuring their correct behavior. The reason for these restrictions are related to two problems: the multipath routing solution being infinite and the occurrence of forward loops. Both of the problems are strictly related to the presence of closed circuits in the network.

A trade-off was clearly identified between the number of elements (which can go from a simple metric to a more complex term, commonly known as policies) presented in the set model that maintain preference (sets L_u) and the existence of more problematic circuits in the network. More policies that maintain preference with the addition of links mean more equally preferred paths and more multipath but they lead to more cycles being problematic and therefore a more careful $\otimes$ operation is necessary. A compromise has to be found and some path compositions need to be considered invalid in order to get a good balance between the level of multipath and the restrictions needed on the network. Correctness without any restrictions on the network can also be achieved with a more restrict condition that forbids label pair elements to belong to the same set of preference maintaining policies.

This work resulted in the implementation of a set of software tools that allows the automatic verification of these results to a particular routing model, it also allows to quickly simulate the routing solution for a particular topology as well as to automatically label topologies to analyze what is the best labeling for a particular case. Therefore, it could be interesting to formalize, as an optimizing problem, the labeling of networks topologies in future works.

In a set of experiments using these software tools we were able to quickly verify how the use of more or less policies as well as the option to have network labeling restrictions or not has an impact on the number of available paths in the routing solution that can be uses for forwarding packets.

Path diversity provides a possibility to a better network traffic control and fault tolerance, as well as facilitating the application of TE, by exploiting the multiple alternatives paths (distributing traffic without changing the routing table).

A multipath policy based protocol offers some improvements. More paths can be available than in an ECMP solution, and forwarding is loop free, with simple destination based forwarding something that is not possible in UCMP (un-equal cost multipath). The use of destination based hop-by-hop forwarding also suppresses the pre-signaled paths needed in the multipath MPLS solution.

For all these reasons, understanding under which conditions a routing protocol can provide multipath policy routing especially maintaining simple destination based distributed forwarding is valuable to design new protocols. This is particularly true when new trends like SDN provide an easier scenario to deploy new routing algorithms. The software build in this work proved to be a valuable tool in the study of particular models and can serve as a tool do develop concrete protocols.

Through the use of the developed tools this work provides an insight on the fundamental limits and trade-off's involved in these types of protocols. With its use, it is also proven that these protocols still present a correct operation with different amounts of path diversity for the same correct operation model restrictions, depending on some decisions on the policy to apply and where to apply. In sum, the tool shows the practical implications of implementing a multipath routing protocol that goes beyond the models used so far.

Bibliography

[1] Pedro Amaral. Multipath Inter-domain Policy Routing. Phd thesis, Universidade Nova de Lisboa, 2012.

[2] C. Cheng, R. Riley, S. P. R. Kumar, and J. J. Garcia-Luna-Aceves. A loop-free extended Bellman-Ford routing protocol without bouncing effect, volume 19. ACM, New York, NY, USA, September 1989.

[3] Andrew S. Tanenbaum and David J. Wetherall. Computer Networks. Prentice Hall Press, Upper Saddle River, NJ, USA, 5th edition, 2010.

[4] J. Doyle and J.D.H. Carroll. Routing TCP/IP 1. CCIE Professional CCIE CCIE Professional Development Series. Cisco, Pearson Education, 2005.

[5] G. Malkin. RIP Version 2. RFC 2453 (Standard), November 1998. Updated by RFC 4822.

[6] Edsger. W. Dijkstra. A note on two problems in connections with graphs. NumerischeMathematik, 1:269–271, 1959.

[7] W. Zaumen and J. J. Garcia-Luna-Aceves. Dynamics of link-state and loop-free distance vector routing algorithms, 1992.

[8] D. Oran. OSI IS-IS Intra-domain Routing Protocol. RFC 1142 (Informational), February 1990.

[9] John Moy. Ospf version 2. Internet RFC 2328, April 1998.

[10] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol Label Switching Architecture. RFC 3031 (Proposed Standard), January 2001.

[11] – Cisco Systems, Inc. BGP/MPLS VPNs. RFC 2547 (Informational), March 1999

[12] – Ixia, Border Gateway Protocol (BGP) Conformance and Performance. White Paper, 2004

[13] G Lee and J Choi. A survey of multipath routing for traffic engineering. Information and Communications University, Korea, 2002.

[14] Byrav Ramamurthy, George Rouskas, and Krishna Sivalingam(eds.). Next-Generation

Internet Architectures and Protocols. Cambridge University Press, 2011.

[15] I. Castineyra, N. Chiappa, and M. Steenstrup. The nimrod routing architecture. RFC 1992, August 1996.

[16] Paolo Narvaez, K-Y Siu, and H-Y Tzeng. Efficient algorithms for multi-path link-state routing. ISCOM'99, November 1999.

[17] SrinivasVutukury and JJ Garcia-Luna-Aceves. A simple approximation to minimum delay routing, volume 29. ACM, 1999.

[18] - C. Hopps. Multipath Issues in Unicast and Multicast Next-Hop Selection. RFC 2991 (Informational), November 2000.

[19] C. Hopps. Analysis of an Equal-Cost Multi-Path Algorithm. RFC 2992 (Informational), November 2000.

[20] Byrav Ramamurthy, George Rouskas, and Krishna Sivalingam (eds.). Next-Generation Internet Architectures and Protocols. Cambridge University Press, 2011.

[21] Wen Xu and Jennifer Rexford. MIRO: multi-path inter-domain routing, volume 36. ACM, 2006.

[22] Yong Liao, LixinGao, Roch Guerin, and Zhi-Li Zhang. Reliable inter-domain routing through multiple complementary routing processes. In Proceedings of the 2008 ACMCoNEXT conference, page 68. ACM, 2008.

[23] Nate Kushman, SrikanthKandula, Dina Katabi, and Bruce M Maggs. R-bgp: Stayingconnected in a connected world. USENIX, 2007.

[24] Xiaowei Yang, David Clark, and Arthur W Berger. Nira: a new inter-domain routing architecture. Networking, IEEE/ACM Transactions on, 15(4):775–788, 2007.

[25] P Godfrey, Igor Ganichev, Scott Shenker, and Ion Stoica. Pathlet routing. ACM SIGCOMM Computer Communication Review, 39(4):111–122, 2009.

[26] MurtazaMotiwala, Megan Elmore, Nick Feamster, and Santosh Vempala. Path splicing. ACM SIGCOMM Computer Communication Review, 38(4):27–38, 2008.

[27] John Scudder, Alvaro Retana, Daniel Walton, and Enke Chen. Advertisement of multiple paths in BGP, Juniper Networks - Network Working Group, October 16, 2013

[28] Igor Ganichev. Inter-domain Multipath Routing. PhD thesis, University of California, Berkeley, Technical Report No. UCB/EECS-2011-136, December 15, 2011.

[29] Francisco Ganhão. Multi-region routing, FCT: DEE – Dissertações de Mestrado, FCT-UNL, 2009.

[30] J. L. Sobrinho, “An algebraic theory of dynamic network routing,” IEEE/ACM Trans. Netw., vol. 13, no. 5, pp. 1160–1173, 2005.

[31] J. Sobrinho, “Algebra and algorithms for QOS path computation and hop-by-hop routing in the internet,” in INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, vol. 2, 2001, pp. 727 –735 vol.2.

[32] T. Griffin. . . , “Increasing bisemigroups and algebraic routing,” Relations and Kleene Algebra in Computer Science, vol. 4988/2008, pp. 123–137, 2008.

[33] T. G. Griffin, F. B. Shepherd, and G. Wilfong, “The stable paths problem and inter-domain routing,” IEEE/ACM Trans. Netw., vol. 10, no. 2, pp. 232–243, 2002.

[34] A. Gurney, “Construction and verification of routing algebras,” Ph.D. dissertation, University of Cambridge, 2009.

[35] M. Gondran and M. Minoux, Graphs, diodes and semirings: new models and algorithms, 2008.

[37] John S Baras and George Theodorakopoulos. Path problems in networks. Synthesis Lectures on Communication Networks, 3(1):1–77, 2010.

[38] Michel Gondran and Michel Minoux. Graphs, diodes and semirings: new models and algorithms, volume 41. Springer, 2008.

[39] J. Sobrinho and T. Griffin, “Routing in equilibrium,” Mathematical Theory of Networks and Systems MTNS, Jan 2010.

[40] T. Griffin, “The stratified shortest-paths problem (invited paper),” Communication Systems and Networks, Jan 2010.

[41] AJT Gurney. Construction and verification of routing algebras. PhD thesis, University of Cambridge, 2009.

[42] C Chau, R Gibbens, and TG Griffin. Towards a unified theory of policy-based routing. Proceedings of IEEE INFOCOM, 2006.

[43] S. H. Strogatz, “Exploring complex networks”, Nature, vol. 410, pp.268-276, March 2001.

[44] R. Albert and A-L. Barabási, “Statistical mechanics of complex networks”, Review of Modern Physics, vol. 74, pp. 47-91, January 2002.

[45] S. Milgram, “The small-world problem”, Psychology Today, vol. 2, pp.60-67, 1967

[46] M. E. J. Newman and D. J. Watts, “Renormalization group analysis of the small-world network model”, Phys. Lett. A, vol. 263, pp. 341-346, 1999.

[47] Marcelo Yannuzzi, Xavier Masip-Bruin, “Open issues in Interdomain Routing: a survey”, Network, IEEE, Vol. 19, Issue: 6, 21 November 2005.

[48] Kyoung, Hee. “Network Measures and Network Models?” CNS-classes.bu.edu. CN710, 10 Sep. 2007. Web. 24 Sep. 2007.

[49] A. Gurney and T. Griffin, “Lexicographic products in metarouting,” in Network Protocols, 2007. ICNP 2007. IEEE International Conference on, oct. 2007, pp. 113 –122.